UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I
CENTRO DE CIÊNCIAS E TECNOLOGIAS
DEPARTAMENTO DE ESTATÍSTICA
CURSO DE BACHARELADO EM ESTATÍSTICA

GIULLBER VALENTIM DA SILVA

CLASSIFICAÇÃO DE IMAGENS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS ATRAVÉS DO TRANSFER LEARNING

CAMPINA GRANDE - PB
2023

GIULLBER VALENTIM DA SILVA

CLASSIFICAÇÃO DE IMAGENS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS ATRAVÉS DO TRANSFER LEARNING

Trabalho de Conclusão de Curso apresentada ao curso de Bacharelado em Estatística do Departamento de Estatística do Centro de Ciências e Tecnologias da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Estatística.

Orientador: Prof. Dr. Tiago Almeida de Oliveira.

CAMPINA GRANDE - PB
2023

É expressamente proibido a comercialização deste documento, tanto na forma impressa como eletrônica. Sua reprodução total ou parcial é permitida exclusivamente para fins acadêmicos e científicos, desde que na reprodução figure a identificação do autor, título, instituição e ano do trabalho.

S586c Silva, Giullber Valentim da.

Classificação de imagens utilizando redes neurais convolucionais através do *transfer learning* [manuscrito] / Giullber Valentim da Silva. - 2023.

67 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Estatística) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2023.

"Orientação : Prof. Dr. Tiago Almeida de Oliveira, Coordenação do Curso de Estatística - CCT. "

1. Pneumonia. 2. Estatística. 3. Redes neurais. I. Título

21. ed. CDD 519.5

Elaborada por Geovani S. de Oliveira - CRB - 15/1009

Biblioteca Central BC/UEPB

GIULLBER VALENTIM DA SILVA

CLASSIFICAÇÃO DE IMAGENS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS ATRAVÉS DO TRANSFER LEARNING

Trabalho de Conclusão de Curso apresentada ao curso de Bacharelado em Estatística do Departamento de Estatística do Centro de Ciências e Tecnologias da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Estatística.

Aprovada em: 24/11/2023.

BANCA EXAMINADORA

Prof. Dr. Tiago Almeida de Oliveira (Orientador)
Universidade Estadual da Paraíba (UEPB)

Prof. Dr. Crysttian Arantes Paixão
Universidade Federal da Bahia (UFBA)

Profa. Ma. Luciana de Queiroz Leal Gomes
Universidade Estadual da Paraíba (UEPB)

A Deus, a minha mãe e familiares e aos meus amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus por todas as bençãos e graças que me concedeu desde que nasci e durante essa caminhada dentro da Universidade.

Agradeço a toda minha família, principalmente a minha mãe, Martineide Valentim Gomes, mesmo trabalhando arduamente sempre me apoiou e me proporcionou as condições necessárias para que eu seguisse meus sonhos e objetivos.

Agradeço também a todas as pessoas que de alguma forma estiveram ao meu lado nessa jornada me apoiando, mesmo nos momentos mais estressantes e tristes, mas sempre estiveram lá como Danyella Elyca Dantas Gomes, minha tia Cícera Valentim e outros que hoje se encontram em meu coração.

Também agradeço as melhores pessoas que alguém poderia fazer e ter como amigos, Gabriel Messias Santana Peixoto, Mateus Silva Rocha, que me ajudaram e me apoiaram durante a escrita deste trabalho. Além de José Lucas, Samuel Souza, Jiulia Feliciana, Suziane, Anderson Severino, José Wellington e dentre outros que compartilharam comigo essa caminha árdua, porém, vitoriosa.

Agradeço também a todos os professores da UEPB que um dia eu tive a oportunidade de ser aluno, de forma especial agradeço ao professor Dr. Ricardo Alves de Olinda por ter aberto as portas para mim no projeto de extensão e ter sido meu orientador de PIBIC. Também agradeço do fundo do coração a professora Dr. Ana Patrícia, ao professores Me. Cleanderson, Dr. Josemir Ramos e dentre outros que sempre me incentivaram a seguir em frente, estudando e nunca desistir de conquistar meus objetivos.

Agradeço também em especial ao meu orientador, professor Dr. Tiago Almeida de Oliveira, que me aceitou como orientando e se dispôs a construir esse trabalho que marca o final da minha jornada na UEPB de forma excepcional.

E por fim, a todos as demais pessoas que conheci, que trabalham na instituição, que sei que de alguma forma contribuíram para a minha formação, o meu mais sincero muito obrigado, por tudo!

“O cérebro se adapta as estatísticas do mundo exterior que maximizam a nossa sobrevivência.”

(Miguel Nicolelis)

RESUMO

Este trabalho aborda a detecção de pneumonia, uma patologia pulmonar e a maior responsável pela morte de crianças ao redor do mundo, a partir de imagens de radiografias pediátricas. O diagnóstico tradicional envolve a avaliação do histórico do paciente assim como radiografias torácicas. Para aprimorar esse processo, este trabalho utiliza modelos classificadores baseados em redes neurais convolucionais dentro da área de *deep learning*. Se utilizando do método de *transfer learning* foi possível utilizar os pesos pré-treinados da Inception-ResNet-V2, uma rede neural convolucional mais robusta, e a VGG-16 que possui uma estrutura bem mais simples. O objetivo é classificar imagens radiográficas infantis para identificar a presença ou ausência de pneumonia, comparando ambos os modelos para identificar o melhor deles, verificando a eficácia do data augmentation no balanceamento da base de imagens de treinamento, além da comparação com outros modelos presentes na literatura. O banco de dados inclui imagens divididas em duas classes sendo a Pneumonia, com 4273 imagens, e a Normal, com 1583 imagens. A base de treinamento, inicialmente desbalanceada, contém 3883 imagens de Pneumonia e 1349 imagens Normais. Para o processo de balanceamento, foi aplicado o método de aumento de dados usando a biblioteca Albumentations em Python. A avaliação dos modelos treinados incluiu métricas como Precision, Recall, F1-score e Acurácia, utilizando a matriz de confusão. Métodos gráficos de avaliação, como a curva ROC e Precision-Recall, também foram empregados. A partir do treinamento inicial com a base desbalanceada foi evidenciado um desempenho semelhante entre os modelos, com ligeira vantagem para a Inception-ResNet-V2 com AUC de 93% e acurácia de 93,26%. No entanto, ao equilibrar a base de treinamento, a VGG-16 apresentou melhorias significativas, atingindo AUC de 94% e acurácia de 94,23%. Na comparação com outros dois modelos existentes na literatura, treinados em 100 épocas, o melhor modelo deste trabalho obteve resultados semelhantes, porém, mais equilibrado quanto a sua sensibilidade e especificidade, sendo treinado em apenas 10 épocas. Com isso, a VGG-16, treinada com uma base balanceada, demonstrou melhor desempenho e equilíbrio na classificação. Quanto a eficácia do *data augmentation* foi mais evidente na VGG-16, indicando que arquiteturas mais simples podem se beneficiar mais desse método em comparação com modelos complexos como a Inception-ResNet-V2, destacando a importância do balanceamento e da escolha adequada de técnicas em diferentes arquiteturas.

Palavras-Chave: Deep Learning; Radiografia Pediátrica; Pneumonia; Data Augmentation.

ABSTRACT

This work addresses the detection of pneumonia, a pulmonary pathology and the leading cause of death in children worldwide, based on pediatric chest X-ray images. The traditional diagnosis involves patient history evaluation along with chest X-rays. To enhance this process, this study employs classifier models based on convolutional neural networks within the field of deep learning. Utilizing the transfer learning method, pretrained weights from Inception-ResNet-V2, a more robust convolutional neural network, and VGG-16, which has a simpler structure, were used. The goal is to classify pediatric X-ray images to identify the presence or absence of pneumonia, comparing both models to determine the superior one. The effectiveness of data augmentation in balancing the training image database is also examined, along with a comparison with other models found in the literature. The database includes images divided into two classes: Pneumonia, with 4273 images, and Normal, with 1583 images. The initially imbalanced training set contains 3883 Pneumonia images and 1349 Normal images. For the balancing process, the data augmentation method using the Albumentations library in Python was applied. Evaluation of the trained models included metrics such as Precision, Recall, F1-score, and Accuracy, using the confusion matrix. Graphical evaluation methods, such as the ROC curve and Precision-Recall, were also employed. From the initial training with the imbalanced dataset, similar performance was observed between the models, with a slight advantage for Inception-ResNet-V2 with an AUC of 93% and an accuracy of 93.26%. However, upon balancing the training set, VGG-16 showed significant improvements, achieving an AUC of 94% and an accuracy of 94.23%. In comparison with two other models existing in the literature, trained for 100 epochs, the best model in this study achieved similar results but demonstrated better balance in terms of sensitivity and specificity, being trained in only 10 epochs. As a result, VGG-16, trained with a balanced dataset, demonstrated better performance and balance in classification. The efficacy of data augmentation was more evident in VGG-16, indicating that simpler architectures may benefit more from this method compared to complex models like Inception-ResNet-V2, underscoring the importance of balance and the appropriate choice of techniques in different architectures.

Keywords: Deep Learning; Pediatric Radiography; Pneumonia; Data Augmentation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	Average Precision (Precisão Média)
AUC	Area Under Curve (Área Sob a Curva)
CNN	Convolutional Neural Network (Rede Neural Convolucional)
FN	Falso Negativo
FP	Falso Positivo
GAP	Global Average Pooling (Agregação Média Global)
MCP	McCulloch - Pitts
MLP	Multilayer Perceprton (Perceptron Multicamadas)
PR	Precision – Recall (Precisão – Sensibilidade)
ReLU	Rectified Linear Unit (Unidade Linear Retificada)
RNA	Redes Neurais Artificiais
ROC	Receiver Operating Characteristic (Característica de Operação do Receptor)
TFP	Taxa de Falsos Positivos
TVN	Taxa de Verdadeiros Negativos
TVP	Taxa de Verdadeiros Positivos
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos

SUMÁRIO

1 INTRODUÇÃO

A pneumonia é considerada uma das principais causas de mortes de crianças com menos de 5 anos de idade em todo o mundo, em 2017 foi constatado que esta doença foi responsável pelo óbito de mais de 808 mil crianças, enquanto em 2019, houve uma diminuição para um pouco mais de 740 mil crianças em todo mundo (WORLD HEALTH ORGANIZATION, 2019, 2022). Esta doença que afeta principalmente o pulmão do infectado, pode ser causada tanto por bactérias, vírus ou até mesmo fungos, abrangendo sintomas como tosse, febre e até mesmo falta de ar (KORKMAZ e TRABER, 2023).

Uma das formas de realizar o diagnóstico é a partir da avaliação do histórico do paciente por um médico, além, da utilização de radiografias do tórax do infectados. Também conhecido como imagens de raio-x, essas radiografias são exames de imagem utilizados na medicina para a detecção de possíveis anomalias pulmonares. Além disso, podem ser utilizadas para revelar possíveis problemas no coração, ossos ou em grandes vasos sanguíneos (RAHMAT , ISMAIL e ALIMAN, 2018).

Sendo assim, para auxiliar e agilizar o processo de diagnóstico utilizando este exame médico, através de imagens de raio-x, pode-se aplicar métodos baseados em visão computacional com a tarefa de realizar classificação de imagens utilizando arquiteturas de aprendizado profundo, também conhecido como *deep learning*.

O aprendizado profundo se baseia na utilização de redes neurais artificiais para a realização de tarefas especificas como a identificação padrões em uma imagem em um processo de classificação (TISSOT, CAMARGO e POZO, 2012).

O uso do *deep learning* vem se tornando bastante comum na área médica por gerar resultados bastantes satisfatórios na tarefa de classificação de imagens, com destaque para o uso das arquiteturas de redes neurais convolucionais que fazem o processo de extração de características da imagem, preservando-as neste processo de filtragem, que torna esta rede neural extremamente importante na análise de imagens de radiografias do tórax (RAHMAT , ISMAIL e ALIMAN, 2018).

Para a utilização de redes neurais convolucionais, geralmente, se emprega a técnica do *transfer learning* que faz a transferência de aprendizagem de uma rede neural já pré-treinada para que seja utilizada em algum processo de classificação. Esta técnica é bastante útil, principalmente, quando a quantidade de dados a serem utilizados não estão em uma quantidade ideal para o treinamento de um rede neural do zero (VOGADO, VERAS, *et al.*, 2019).

Contudo, uma situação bastante recorrente na abordagem do aprendizado de máquinas é a escassez de dados, principalmente na área médica que geralmente dependem de dados ou imagens advindos de exames reais. Essa escassez pode gerar problemas no processo de treinamento dos modelos de aprendizado de máquina, tornando-os mais suscetíveis a *overfitting* (SHORTEN e KHOSHGOFTAAR, 2019).

Porém, este problema pode ser mitigado utilizando técnicas de aumento artificial de dados, como o *data augmentation* que é um método que irá inflar artificialmente uma base de imagens com transformações geométricas definidas a partir das imagens existentes (SHORTEN e KHOSHGOFTAAR, 2019).

Com isso, o objetivo deste estudo é a aplicação de duas redes neurais convolucionais, a partir da utilização do método do *transfer learning*, para a realização da classificação de radiografias infantis na detecção ou ausência de pneumonia, comparando os resultados e avaliando qual destas arquiteturas apresentaram melhor desempenho. Estes modelos foram treinados utilizando a base de imagens de treinamento balanceada pelo método do *data augmentation* e comparadas entre si para avaliar assim a eficácia da aplicação deste método em relação aos treinados na base desbalanceada.

Além disso, utilizando o modelo que demonstrou melhor desempenho neste estudo, realizou-se uma comparação com outros modelos encontrados na literatura, que empregaram a mesma base de imagens de radiografias pediátricas. Essa análise visa avaliar a eficácia do melhor modelo deste estudo em comparação com outros existentes, fornecendo uma perspectiva abrangente sobre seu desempenho em relação às pesquisas anteriores nessa área.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Estrutura de uma Imagem Digital

Uma imagem digital é formada por números dispostos matricialmente onde cada elemento dessa matriz representa um valor individual de um *pixel* da imagem. Portanto, a combinação de milhares destes *pixels* resultará na formação de uma imagem digital completa. A resolução de uma imagem é diretamente ligada a quantidade de *pixels*, quanto maior for a quantidade destes, maior será a resolução da imagem, fornecendo, assim, uma estrutura organizada para processos como visualização e manipulação das imagens (ANDRADE, 2020).

Um *pixel* é basicamente a menor informação que forma uma imagem e uma unidade de discretização do ambiente contínuo do mundo real para, assim, ser representado de forma digital. Esse processo de tornar discreto uma informação contínua do mundo real é chamado de rasterização, este processo vincula a cada região contínua um valor numérico discreto chamado de *pixel* e com isso se dá a criação de imagens *bitmap* (ALMEIDA e MAGRINI, 2021).

Dessa forma, uma imagem *bitmap*, que traduzindo seria um mapa de bits, é formada por conjuntos de pontos de *pixels*, sendo que cada *pixel* é composto por 8 *bits* com a cor da imagem sendo definida pela quantidade de *bits* utilizados para representá-la. Podemos então definir três grupos de imagem quanto a sua escala de cores, as binárias que assumirão apenas as cores preto e branco, as em escala de cinza que em cada *pixel* representará uma cor na escala de cinza e, por fim, as coloridas, geralmente, se utilizando do sistema de cores RGB[1] (Blue – B) (ANDRADE, 2020).

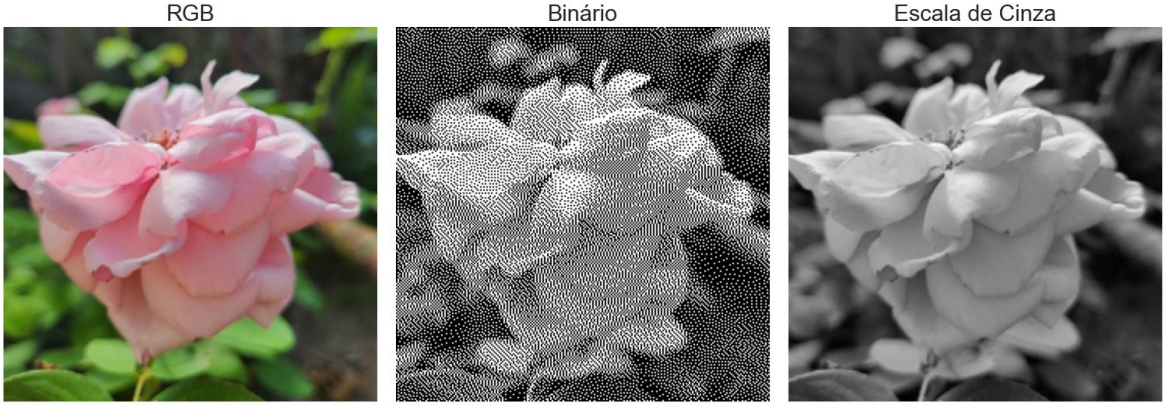
O número de cores que um *pixel* pode representar é definido por $\mathbf{2^n}$, onde $\boldsymbol{n}$ é a quantidade de *bits*, por exemplo, um *pixel* com 8 *bits* poderá representar $\mathbf{2^8 = 256}$ tonalidades cores, variando em um intervalo de $[\mathbf{0}, \mathbf{255}]$. Em uma imagem em formato de escala de cinza quanto mais próximo o valor do *pixel* for de 0 menos cinza, assumindo uma cor mais próxima do preto, será este ponto na imagem, por outro lado, quanto mais próximo de 255 mais cinza este ponto será (ANDRADE, 2020).

Já em imagens coloridas que se utilizam do sistema RGB também existirá 256 tonalidades de cores, porém, essas tonalidades serão referentes a cada um dos canais de cores

[1] No sistema RGB as cores são definidas com base na quantidade de vermelho (Red – R), verde (Green – G) e azul.

que existem neste sistema, portanto, a quantidade de cores que um *pixel* pode assumir neste caso será dado por $\mathbf{256^3 = 16.777.216}$ (ALMEIDA e MAGRINI, 2021)**.**

Figura 1 – Exemplo de imagens colorida (esquerda), binária (meio) e na escala de cinza (direita).



Fonte: Elaborado pelo autor, 2023.

Com base nisso, Almeida e Magrini (2021) definem que a forma matricial de uma imagem digital pode ser dado por uma matriz $\boldsymbol{A_{n\times m}}$ onde cada elemento $\boldsymbol{a_{ij}}$ desta matriz será associado a um valor numérico, pertencente ao conjunto do números naturais, de um *pixel*, sendo assim, cada elemento desta matriz será a representação da tonalidade da cor.

Quando é realizado algum processo de transformação na imagem, seja ela uma rotação, recorte ou mudança de brilho e contraste, o que estará sendo modificado é justamente os valores ou posições dos *pixels* na matriz $\boldsymbol{A_{n\times m}}$(ANDRADE, 2020)**.**

$$\boldsymbol{A_{n\times m}} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \tag{2.1}$$

Portanto, podemos escrever como notação de dimensão de *pixels* de uma imagem digital usando $(\boldsymbol{n \times m})$. Por exemplo, considerando uma imagem binária cuja dimensões de *pixels* é de $(\mathbf{5 \times 5})$, ou seja, teremos uma matriz $\boldsymbol{A_{5\times 5}}$ em que cada elemento desta matriz será igual a **1** se a cor do *pixel* for branco e **0** se a cor for preta (Figura 2).

Figura 2 – Exemplo utilizando uma imagem binária de dimensões (5 x5) representando o número 9 e sua respectiva matriz de *pixels*.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Fonte: Almeida e Magrini (2021)

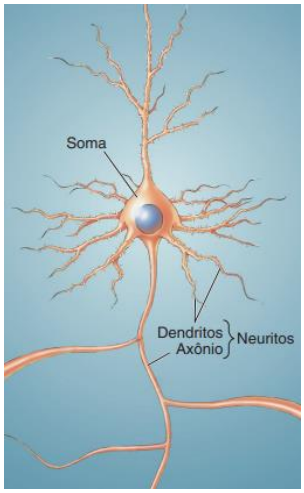
2.2 Redes Neurais Biológicas

O Cérebro humano é uma estrutura biológica, altamente complexa, responsável por controlar as funções sensório motoras e autônomas do corpo humano. Possui uma capacidade poderosa em reconhecer padrões, na aprendizagem obtida por experiência, além de conseguir realizar interpretações daquilo que se observa (BRAGA, CARVALHO e LUDERMIR, 2000).

Se tratando de redes neurais biológicas, como sendo um conjunto de neurônios interligados entre si, a estimativa para a quantidade destas redes é em torno de 500 para cerca de 1000 módulos principais, cada rede neural possui uma quantidade de neurônios estimada em 100.000, totalizando uma quantidade geral de neurônios na ordem de 10 a 500 bilhões (EBERHART e DOBBINS, 1990).

A partir da aplicação do procedimento de Golgi, um método em que se impregna uma determinada amostra do tecido neural com uma solução de cromato de prata, pode-se identificar e classificar o neurônio em três partes principais: soma, dendritos e axônio (Figura 3) (BEAR, CONNORS e PARADISO, 2017). A soma de um neurônio é o corpo da célula onde será processada as informações advindas de outros neurônios recebidas pelos dendritos e o resultado desse processo é transmitidos para outros neurônios através do axônio (BRAGA, CARVALHO e LUDERMIR, 2000).

Figura 3 – As partes principais de um neurônio.

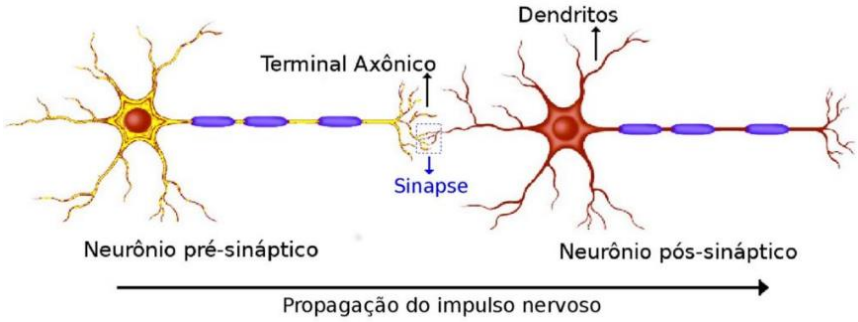


Fonte: Bear, Connors e Paradiso (2017).

Com isso, os autores Bear, Connors e Paradiso (2017) definem uma terminação axonal como sendo um local em que um axônio de um neurônio entra em contato com outros neurônios, ou outras células, e passa as devidas informações. Este ponto de interação é chamado de sinapse que se configura em pré-sináptico quando é correspondente ao lado da terminação axonal e o pós-sináptico que corresponde ao lado onde possa estar um dendrito ou soma de outro neurônio (Figura 4).

Os impulsos elétricos que carregam as informações transmitidas por um certo neurônio ao longo do axônio, ao chegarem em uma terminação axonal, serão convertidos em sinais químicos e, assim que forem absorvidos pelos receptores do dendrito, serão convertidos novamente em impulsos elétricos e enviados até a soma do neurônio receptor (BEAR, CONNORS e PARADISO, 2017).

Figura 4 – Ilustração do processo sináptico entre dois neurônios biológicos.



Fonte: Flores (2020)

Estes novos impulsos elétricos, gerados pelo processo pós-sináptico, ao serem recebidos pelo corpo do neurônio receptor serão comparados com os demais sinais recebidos e se este percentual de carga for relativamente alto, excedendo o seu limiar de excitação, o neurônio receptor irá disparar e transmitirá essa nova informação para os demais na rede neural (BRAGA, CARVALHO e LUDERMIR, 2000).

2.3 Rede Neurais Artificiais

As redes neurais artificiais são inspiradas no esquema de conexões neuronais biológicas do cérebro humano utilizando técnicas de aprendizagem computacional, a partir de modelos matemáticos, capazes de realizar tarefas complexas com base em uma combinação de elementos básicos (MARTINS, 2019). Com isso, a forma em que o conhecimento é adquirido por uma Rede Neural Artificial (RNA) se dá através de um processo de aprendizagem e pelos pesos sinápticos que são utilizados como uma forma de armazenamento do conhecimento adquirido, se assemelhando a estrutura neural biológica (HAYKIN, 2001).

Podemos ainda definir uma rede neural como sendo um tipo de máquina projetada com a finalidade de reproduzir a forma em que o cérebro humano realiza tarefas de interesse, interligadas por células computacionais denominadas de neurônios (nodos ou unidades de processamento) e que possui uma predisposição de forma natural em armazenar o conhecimento adquirido e disponibilizá-lo para o uso (HAYKIN, 2001).

Uma das principais características de uma RNA é a sua capacidade de conseguir identificar e apresentar respostas em relação a algum padrão desconhecido ou que não tenha sido informado no seu processo de treinamento (NIED, 2007). Esse poder de generalização é influenciado tanto pela sua topologia quanto pela utilização de algum algoritmo de aprendizagem (HAYKIN, 2001).

O primeiro trabalho que deu início e base para os estudos sobre RNA foi publicado em 1943 pelos autores Warren S. McCulloch e Walter Pitts, onde tentaram reproduzir o funcionamento de um neurônio biológico a partir de um modelo matemático. Assim, McCulloch e Pitts (1943) criaram 5 suposições para descreverem o funcionamento de um neurônio:

> 1 – A atividade do neurônio é um processo de “tudo ou nada”; 2 – Um neurônio fixo de sinapses deve ser excitado dentro de um período de adição latente para excitar um neurônio a qualquer momento, e esse número é independente da atividade anterior e da posição no neurônio; 3 – O único atraso significativo dentro do sistema nervoso é o atraso sináptico; 4 – A atividade de qualquer sinapse inibitória impede

> absolutamente a excitação do neurônio naquele momento; 5 – A estrutura da rede não muda com o tempo;

Portanto, essas suposições foram responsáveis pela criação da ideia do primeiro neurônio artificial chamado de neurônio de McCulloch – Pitts (MCP). Um ponto marcante na história, cujo suas teorias demonstraram a capacidade do neurônio MCP em realizar qualquer expressão lógica (EBERHART e DOBBINS, 1990).

O modelo matemático do MCP é descrito como tendo $\boldsymbol{n}$ entradas $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots, \boldsymbol{x_n}\}$ e com apenas uma saída $\boldsymbol{y}$. Os canais de entradas seriam a representação dos dendritos de um neurônio biológico e enquanto a saída única representaria o axônio. Além disso, com a finalidade de imitar as sinapses foi utilizado o conceito de pesos ligados a cada entrada $\boldsymbol{x_i}$ descritos como $\{\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \dots, \boldsymbol{w_n}\}$, cujo valor pós-sináptico será dado pela soma ponderada das entradas com seus respectivos pesos (BRAGA, CARVALHO e LUDERMIR, 2000).

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i w_i} \tag{2.2}$$

De maneira semelhante, o neurônio MCP também terá a particularidade dita como "tudo ou nada", sendo determinada quando o valor da soma ponderada dada pela equação (2.2) for maior ou igual a um valor de limiar do neurônio, representado por $\boldsymbol{\theta}$ que pode ser verificado na equação (2.3) (BRAGA, CARVALHO e LUDERMIR, 2000).

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i w_i} \geq \boldsymbol{\theta} \tag{2.3}$$

2.3.1 O Perceptron

Utilizando como inspiração o modelo do neurônio MCP, em 1958, Frank Rosenblatt criou o modelo *perceptron* que é composto por uma estrutura de rede e por uma regra de aprendizado, cujo as unidades de processamento são neurônios MCP. Com isso, Rosenblatt foi o primeiro a introduzir o conceito de aprendizagem ligado as redes neurais (BRAGA, CARVALHO e LUDERMIR, 2000).

A estrutura do *perceptron* (Figura 5) é definida em duas camadas: a primeira sendo a camada de entrada, onde cada peso é associado a cada entrada correspondente e definido aleatoriamente e fixo durante o processo de treinamento; a segunda camada é a de saída, onde serão realizados os processos de soma e aplicação de uma função de limiar (NIED, 2007).

Porém, este modelo é conhecido por *perceptron* de camada única, pois apenas a camada de saída é quem possui pesos treináveis.

Como no modelo matemático do neurônio de MCP, cada entrada estará associada a um peso sináptico, cujo a soma ponderada dada pela equação (2.2) resultará em uma combinação linear que iremos denominar de $\boldsymbol{u}$ (RAUBER, 2005). Após o processo de soma ainda será aplicado externamente um bias, dado por $\boldsymbol{b}$, com a tarefa de aumentar ou diminuir o valor da entrada na função de ativação, aplicando assim uma transformação afim na saída do combinador linear do modelo (HAYKIN, 2001).

Logo, Haykin (2001) define o modelo matemático de um *perceptron* por duas equações, em que a primeira é uma combinação linear chamada de função soma com a adição da bias, dada por:

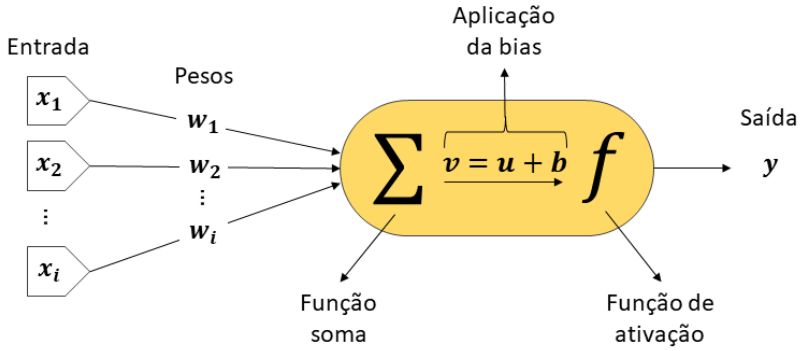
$$\boldsymbol{v = u + b} \tag{2.4}$$

onde o combinador linear será $\boldsymbol{u = \sum_{i=1}^{n} x_i w_i}$, enquanto a segunda equação chamada de função de ativação, resultará no sinal de saída e será dada por:

$$\boldsymbol{y = f(v)} \tag{2.5}$$

Portanto, na Figura 5 está a representação da estrutura de um *perceptron* onde pode-se visualizar desde a entrada dos sinais, a aplicação dos processos matemáticos, a função soma e a função de ativação, até o sinal de saída resultante. Nota-se um processo baseado no aprendizado a diante, em uma direção.

Figura 5 – Estrutura de um *perceptron* de uma única camada.



Fonte: Elaborado pelo autor, 2023.

A aplicação do *perceptron* objetiva a classificação precisa de padrões estimulados externamente pelos valores das entradas em uma de duas classes, linearmente separáveis, dada por $\boldsymbol{\varphi_1}$ quando o valor da saída for +1 ou $\boldsymbol{\varphi_2}$ quando o valor da saída de for -1, sendo definido

a partir da aplicação de uma função de ativação com base no valor de $\boldsymbol{v}$ (HAYKIN, 2001; NIED, 2007).

2.3.1 Multilayer Percetron

A estrutura de uma *multilayer perceptron* (MLP) consiste em uma rede neural composta por vários *perceptrons*, organizados em várias camadas e conectados de forma simples, representando um modelo que possui uma característica não linear entre o vetor de entrada e o vetor de saída (GARDNER e DORLING, 1998; DELASHMIT e MANRY, 2005).

Portanto, diferentemente do *perceptron* de uma única camada, uma MLP é capaz de lidar com problemas não lineares de classificação. Haykin (2001) define que este problema pode ser verificado na aplicação do XOR (*Exclusive OR*), uma operação lógica em que, se considerarmos quatro vértices com padrões de entrada (0,0), (0,1), (1,0) e (1,1), retornará o valor de 1 se o padrão for diferente ou 0 se o padrão forem iguais, este problema resultará em um caso de não linearidade (Figura 6).

O perceptron de uma única camada é capaz de solucionar problemas como as operações lógicas AND (e) e OR (ou), em que se aplicarmos ambas em relação aos mesmo padrões de entradas citados acima, teremos os resultados apresentados na Tabela 1.

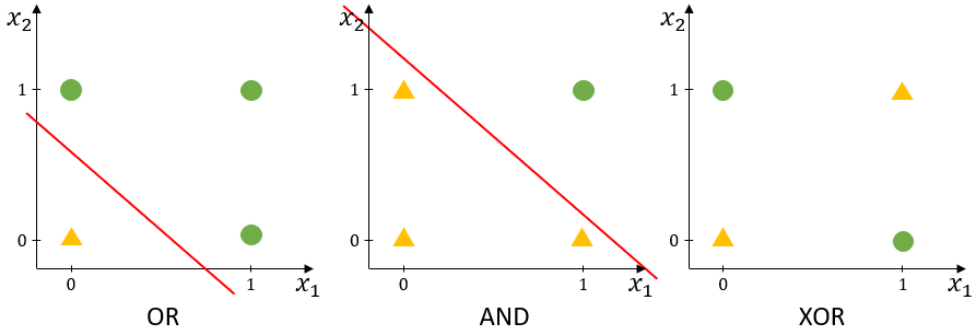
Tabela 1 – Aplicação dos operadores lógicos em relação a um determinado padrão de entrada (x_1, x_2).

x_1	x_2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Fonte: Elaborado pelo autor, 2023.

Este problema pode ser verificado na Figura 6, onde o resultado igual a 1 será representado por um círculo com a cor verde e quando for 0 será um triângulo na cor amarela, além disso. Para os operados AND e OR, que podem ser linearmente separáveis, apresentam uma linha vermelha demonstrando a separação.

Figura 6 – Representação do problema linear em ambos os operadores.



Fonte: Elaborado pelo autor, 2023.

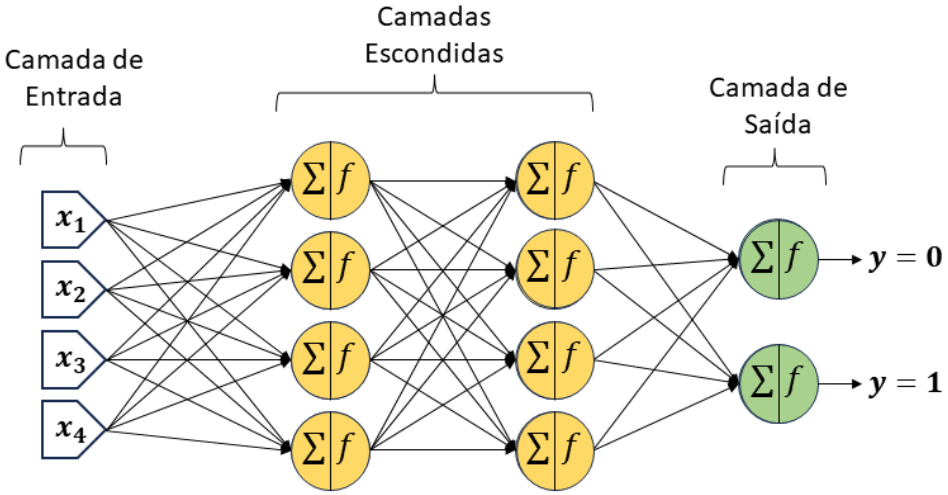
Deste modo, uma forma de contornar este problema de classificação não linear que um *perceptron* não conseguiria realizar, é a aplicação de uma MLP que com suas camadas ocultas compostas por vários neurônios, possibilitam que esta rede neural possa lidar com problemas mais complexos que envolvam o caso do XOR, que gera uma não linearidade (HAYKIN, 2001).

Nesta RNA, o sinal informado na camada de entrada irá se propagar adiante por cada unidade de processamento em cada camada até chegar na última camada em que será retornado um sinal de saída (HAYKIN, 2001). Ou seja, como define Nied (2007), cada unidade de processamento de cada camada da rede terão como entrada os sinais de saída advindos de outras unidades da camada precedente, configurando assim uma rede neural *feedforward*.

Por definição, a estrutura de uma MLP é composta por no mínimo três camadas: as camadas de entrada e saída, com a adição das camadas intermediárias ou também conhecidas como camadas escondidas (NIED, 2007). A quantidade de neurônios utilizados nesse modelo de rede neural dependerá do problema a ser aplicado. No caso da camada de entrada, este número dependerá dos padrões existentes a serem reconhecidos, enquanto para a de saída será relacionado a quantidade de classes que o modelo irá classificar ao final do processo de aprendizagem (RAMCHOUN, 2016).

Esta estrutura totalmente interligada é denominada como Rede Neural Completamente Conectada (Figura 7), em que cada unidade de processamento de uma camada está ligado aos demais da camada posterior e assim por diante (MARTINS, 2019). A estrutura que liga cada unidade de processamento é denominada de aresta e fazem com que as informações sejam transmitidas entre cada unidade de diferentes camadas, que em cada uma destas arestas será incluído um peso $\boldsymbol{w_i}$ que modulará o sinal de saída de uma unidade para outra adjacente (CARDON, MÜLLER e NAVAUX, 1994).

Figura 7 – Representação da estrutura de uma rede neural MLP.



Fonte: Elaborado pelo autor, 2023.

2.3.3 Função de Ativação

Este tipo de função pode ser descrita como responsável pela determinação da forma e da intensidade em que o será o sinal de saída será transmitido (CARDON, MÜLLER e NAVAUX, 1994). Dentre as funções de ativação existentes podemos citar a Função de Limiar que, também conhecida *heaviside* ou função de escada (Figura 8a), Rauber (2005) define como sendo uma função não-linear discreta que irá produzir uma saída binária de acordo com o valor $\boldsymbol{v}$ dada pela equação (2.4), onde podemos descrever a saída $\boldsymbol{y}$ como:

$$\boldsymbol{f(v)} = \begin{cases} \mathbf{1}, \boldsymbol{se\ v \geq 0} \\ \mathbf{0}, \boldsymbol{se\ v < 0} \end{cases} \tag{2.6}$$

A função linear (Figura 8b) que podemos descrever como sendo uma equação linear de forma que:

$$\boldsymbol{f(v) = \alpha v} \tag{2.7}$$

em que o parâmetro $\boldsymbol{\alpha}$ é um número escalar positivo responsável pela determinação da inclinação da reta (CARDON, MÜLLER e NAVAUX, 1994).

Outra função de ativação é a chamada de Função Linear por Partes (Figura 8c), Haykin (2001) a define como sendo uma aproximação de um amplificador não-linear, sendo descrita como:

$$f(v) = \begin{cases} 1, & se\ v \geq +\frac{1}{2} \\ v, se + \frac{1}{2} > v > -\frac{1}{2} \\ 0, & se\ v \leq -\frac{1}{2} \end{cases} \quad (2.8)$$

Por outro lado, a Função Sigmoide (Figura 8d) é mais utilizada de todas as funções de ativação e é descrita como sendo uma função estritamente crescente, exibindo um balanceamento entre o comportamento linear e não-linear de forma adequada (HAYKIN, 2001). Esta função também mapeará a saída em valor binário, sendo dado por:

$$f(v) = \frac{1}{1 + e^{-\alpha v}} \quad (2.9)$$

onde α determinará a inclinação da função.

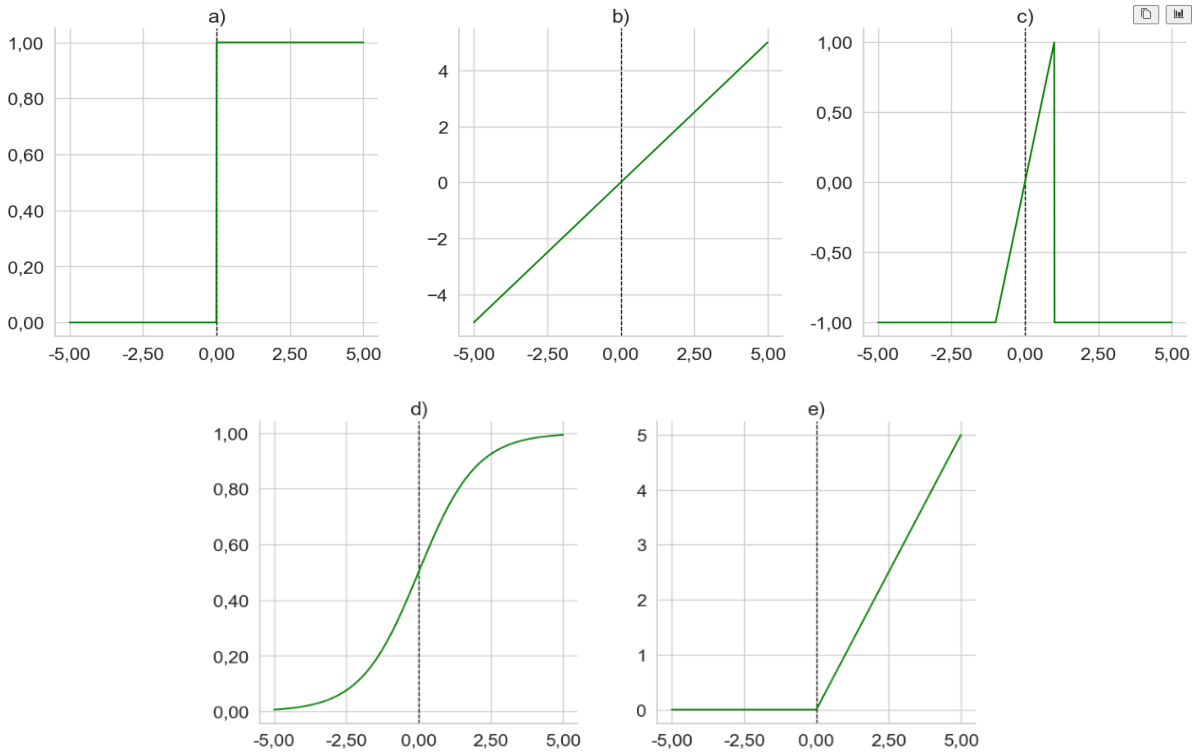
Uma outra função de ativação bastante usada é a Função Softmax, (equação 2.10) cujo seu valor de saída será uma probabilidade de que uma determinada amostra possa pertencer a uma *i-ésima* categoria (Figura 8e) (WANG, LU, *et al.*, 2018). Esta função é mais comumente usada na camada de saída de uma RNA caracterizada por ser uma MLP.

$$f(v) = \frac{e^{v_i}}{\sum_{j=1}^{n} e^{v_j}} \quad (2.10)$$

Por fim, a função ReLU (*Rectified Linear Unit* - Unidade Linear retificada), representada na Figura 8f, é uma das mais utilizadas na construção de RNA por ser simples e possuir uma eficiência computacional significativa na otimização da velocidade de aprendizado dos modelos (IDE e KURITA, 2017). Ela é dada pela equação 2.11 que retorna um valor com base na função linear de entrada se ele for positivo, caso contrário, retornará 0 se este valor for negativo (GÉRON, 2019).

$$f(v) = max(v, 0) \quad (2.11)$$

Figura 8 – Representação gráfica das funções de ativação *heaviside* (a), linear (b), linear por partes (c), sigmoid (d) e a ReLU (e).



Fonte: Elaborado pelo autor, 2023.

2.3.4 Backpropagation

A utilização de uma MLP, modelo que é uma generalização do perceptron, permite utilizar dados não-lineares, o que aumenta o poder de processamento da rede neural (NIED, 2007). Para esse tipo problema, as MLP's são treinadas utilizando um algoritmo de treinamento de retropropagação denominado de *backpropagation*.

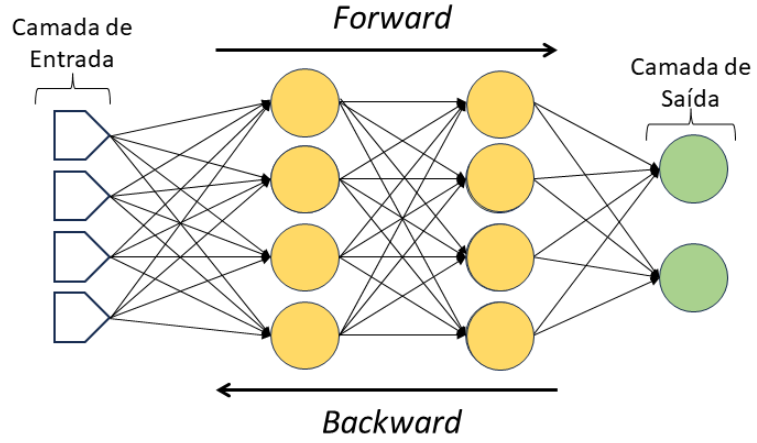
O aprendizado utilizando este algoritmo é realizado através da atualização dos pesos da RNA a partir do cálculo do erro após cada rodada de treinamento. Ou seja, a cada rodada de treinamento será realizado o cálculo do erro através de uma função de perda e os pesos iniciais serão atualizados de maneira que a diferença entre o valor atual da saída e o valor desejável diminua a cada etapa do processo de aprendizagem (TISSOT *et al*., 2012).

Em uma MLP, o sinal de entrada se propaga pela rede em apenas um sentido, sempre para frente, percorrendo cada neurônio artificial de todas as camadas, sendo classificadas como redes neurais *feedforward*. Utilizando o método do retropropagação para a atualização dos pesos, o sinal final, advindo da camada de saída, será comparado com o sinal desejado e resultando no cálculo de um erro associado em que, com isso, ocorrerá o processo inverso

saindo da última camada até a camada de entrada atualizando os pesos de cada arestas da rede neural com base no erro calculado.

Braga, Carvalho e Ludermir (2000) definem estes dois processos como sendo a etapa *forward*, o caminho normal de fluxo do sinal na rede, sempre a frente, enquanto o processo de atualização dos erros sendo chamado de *backward*, ou seja, o sinal de erro se propagando na direção oposta (Figura 9).

Figura 9 – Representação do fluxo do sinal a partir do processo *forward* e da atualização dos pesos dado pelo processo *backward.*



Fonte: Elaborado pelo autor, 2023.

Este algoritmo é baseado na regra delta que define o modo em que os pesos serão atualizados de acordo com a derivada parcial do erro para cada peso. Com isso, podemos chamar o método *backpropagation* como sendo uma regra delta generalizada, em que o cálculo do erro será dado pela soma dos erros quadráticos médios, definida na equação 2.12 (BRAGA, CARVALHO e LUDERMIR, 2000).

$$\boldsymbol{E = \frac{1}{2} \sum_{j=1}^{k} (d_j - y_j)^2}, \tag{2.12}$$

em que $\boldsymbol{d_i}$ será a saída desejada, $\boldsymbol{k}$ o número de neurônios na camada de saída e $\boldsymbol{y_j}$ será o $\boldsymbol{j}$-*ésimo* sinal de saída gerado pela rede, em que aplicando a equação (2.5) em relação a cada neurônio artificial $\boldsymbol{j}$, tem-se que:

$$\boldsymbol{y_j = f_j(v_j)}, \tag{2.13}$$

e, portanto, reescrevendo a equação (2.6) de forma que possamos utilizar apenas o combinador linear $\boldsymbol{u}$, teremos:

$$\boldsymbol{v_j} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i w_{ji}}, \tag{2.14}$$

onde $\boldsymbol{n}$ é a quantidade de conexões de entrada em um determinado neurônio $\boldsymbol{j}$, e ainda $\boldsymbol{w_{ji}}$ o peso na conexão de entrada entre o $\boldsymbol{x_i}$ e o $\boldsymbol{j}$-*ésimo* neurônio.

Contudo, para podermos chegar na equação de ajuste dos pesos é necessário aplicar a regra delta (Δ) em relação ao peso $\boldsymbol{w_{ji}}$ de forma que $\boldsymbol{\Delta w_{ji}}$ seja proporcional a derivada parcial $\frac{\boldsymbol{\partial E}}{\boldsymbol{\partial w_{ji}}}$ (HAYKIN, 2001). Portanto, assim como Braga, Carvalho e Ludermir (2000), podemos descrever a variação dos pesos pelo método do gradiente descendente definido por:

$$\boldsymbol{\Delta w_{ji}} \propto -\frac{\boldsymbol{\partial E}}{\boldsymbol{\partial w_{ji}}}, \tag{2.15}$$

em que o sinal negativo indica o método de descida do gradiente e a partir da aplicação da regra da cadeia podemos calcular o valor da derivada parcial dada na equação (2.15), teremos então:

$$\frac{\boldsymbol{\partial E}}{\boldsymbol{\partial w_{ji}}} = \frac{\boldsymbol{\partial E}}{\boldsymbol{\partial v_j}} \frac{\boldsymbol{\partial v_j}}{\boldsymbol{\partial w_{ji}}}, \tag{2.16}$$

neste caso $\boldsymbol{v_j}$ é dado pela equação (2.14), a derivada $\frac{\boldsymbol{\partial v_j}}{\boldsymbol{\partial w_{ji}}}$ pode ser calculada por:

$$\frac{\boldsymbol{\partial v_j}}{\boldsymbol{\partial w_{ji}}} = \boldsymbol{x_i}, \tag{2.17}$$

ou seja, $\boldsymbol{x_i}$ será o sinal de entrada do $\boldsymbol{j}$-*ésimo* neurônio, ainda temos que a primeira derivada a direita do sinal de igualdade na equação (2.16) é responsável por medir o erro no neurônio $\boldsymbol{j}$, sendo assim, iremos abreviar ela por $\boldsymbol{\delta_j}$ que será o gradiente local, portanto:

$$\boldsymbol{\delta_j} = \frac{\boldsymbol{\partial E}}{\boldsymbol{\partial v_j}} = \frac{\boldsymbol{\partial E}}{\boldsymbol{\partial y_j}} \frac{\boldsymbol{\delta y_i}}{\boldsymbol{\partial v_j}}, \tag{2.18}$$

para a segunda derivada após a segunda igualdade na equação (2.18), pode ser facilmente definida derivando-a em relação a $\boldsymbol{v_j}$:

$$\frac{\boldsymbol{\partial y_i}}{\boldsymbol{\partial v_j}} = \frac{\boldsymbol{\partial f(v_j)}}{\boldsymbol{\partial v_j}} = \boldsymbol{f'(v_j)}, \tag{2.19}$$

já a primeira derivada da segunda igualdade na equação (2.18) pode ser definida de duas formas dependendo da camada em que o neurônio $\boldsymbol{j}$ se encontrará. Se considerarmos que este neurônio se encontra na camada de saída da rede neural, então podemos utilizar para o cálculo do erro $\boldsymbol{E}$ a equação (2.12), sendo:

$$\frac{\partial E}{\partial y_j} = \frac{\partial(\frac{1}{2}\sum_{i=1}^{k}(d_i - y_i)^2)}{\partial y_i} = (d_i - y_i), \tag{2.20}$$

com isso podemos assim definir, para este caso, a fórmula para a determinação do gradiente local δ_j como sendo:

$$\delta_j = (d_i - y_i) f'(v_j). \tag{2.21}$$

Porém, se o neurônio j não for da camada de saída, para calcularmos o valor da derivada $\frac{\partial E}{\partial y_j}$ deve-se utilizar a regra da cadeia, onde:

$$\begin{aligned} \frac{\partial E}{\partial y_j} &= \sum_{l=1}^{M} \frac{\partial E}{\partial v_l} \frac{\partial v_l}{\partial y_j} \\ &= \sum_{l=1}^{M} \frac{\partial E}{\partial v_l} \frac{\partial \sum_{i=1}^{n} w_{il} y_i}{\partial y_j} \\ &= \sum_{l=1}^{M} \frac{\partial E}{\partial v_j} w_{jl} \end{aligned} \tag{2.22}$$

Pela equação (2.17), podemos rescrever o resultado obtido na (2.22) de forma que:

$$\frac{\partial E}{\partial y_j} = \sum_{i=1}^{M} \delta_l w_{jl} \tag{2.23}$$

A equação 2.21 pode ser usada para o cálculo do gradiente local quando o neurônio não estiver na camada de saída, ou seja, quando o j-ésimo neurônio estiver nas camadas intermediárias a fórmula usada será definido por:

$$\delta_j = f'(v_j) \sum_{l} \partial_l w_{jl} \tag{2.24}$$

Portanto, pode-se definir que a fórmula utilizada para realizar o ajuste dos pesos pelo algoritmo *backpropagation* será dado por:

$$\Delta w_{ji} = \eta \delta_j x_i \tag{2.25}$$

em que η é referente a taxa de aprendizagem do algoritmo (HAYKIN, 2001).

2.4 Rede Neurais Convolucionais

Uma Rede Neural Convolucional (*Convolutional Neural Network* - CNN) é um tipo de RNA inspirada na estrutura biológica do córtex visual humano e são extremamente eficazes na extração e aprendizado de características complexas, e hierárquicas por meio de processos realizados em suas camadas de convolução (ALOYSIUS e GEETHA, 2017). Se caracteriza por possuir um processo de aprendizado supervisionado, em que basicamente, se trata de uma generalização de uma MLP para o reconhecimento de formas bidimensionais com alto grau de invariância (HAYKIN, 2001).

Uma CNN é bastante robusta ao analisar e reconhecer características de imagens com variações quanto a processos de rotação, deslocamento, distorção e dentre outros. De forma a garantir este grau de invariância, ela se utiliza de três aplicações nos processos que ocorrem dentro de sua arquitetura: os campos receptivos, os pesos compartilhados e o processo de subamostragem espacial (LECUN *et al*., 1998).

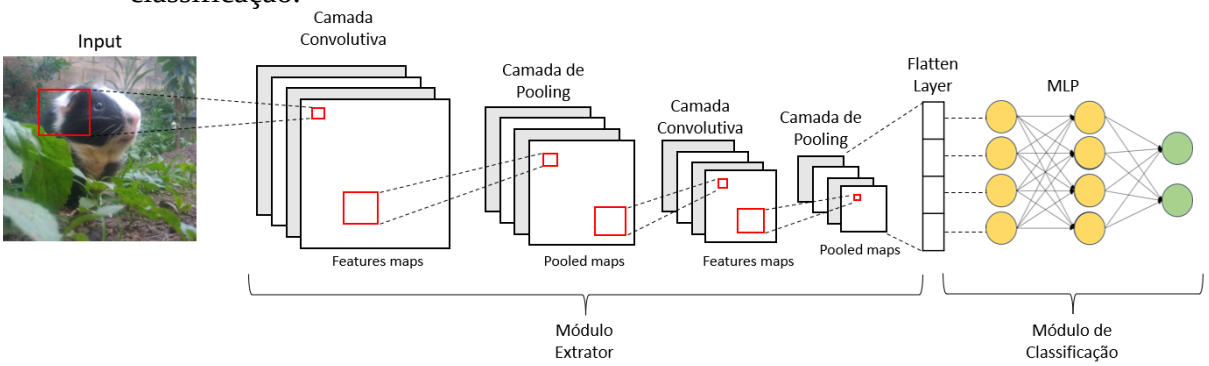
Portanto, como Lecun *et al*. (1998), podemos dividir uma CNN em dois módulos principais sendo, o Módulo Extrator (*Feature Extraction Module*) que será responsável pela extração das principais características da imagem e o Módulo de Classificação (*Trainable Classifier Module*), uma rede neural totalmente conectada, que possuirá a tarefa de realizar o processo de classificação da imagem.

Neste caso, o processo convolucional ocorrerá dentro do módulo extrator de uma CNN, em que estará ligado ao segundo módulo em que será aplicado uma MLP para o processo de classificação. Um dos grandes benefícios de extrair as principais características de uma imagem, por exemplo, é complementar e melhorar o desempenho realizado pelos classificadores, já que são limitados a dados com baixas dimensões (LECUN *et al*., 1998).

Por mais que a CNN seja uma generalização da MLP, existe uma diferença na forma em que cada unidade de processamento está interligada, uma vez que a MLP possui uma estrutura totalmente interligada, já em uma CNN apenas alguns subconjuntos de entradas serão ligados a cada um dos neurônios (VARGAS, PAES e VASCONCELOS, 2016).

Dentro do módulo extrator de uma CNN (Figura 10) existem camadas que realizam o processo de extração das características, onde cada neurônio receberá sinais de entrada de um campo receptivo, o mapeamento das características em que serão alocadas em matrizes onde cada neurônio compartilhará dos mesmos pesos sinápticos e o processo da subamostragem responsável pela diminuição da dimensionalidade da matriz em que foi realizado o mapeamento das características (HAYKIN, 2001).

Figura 10 - Representação de uma rede neural convolucional (CNN) com as camadas do módulo extrator quanto a MLP do módulo de classificação.



Fonte: Elaborado pelo autor, 2023.

2.4.1 Camadas Convolutivas

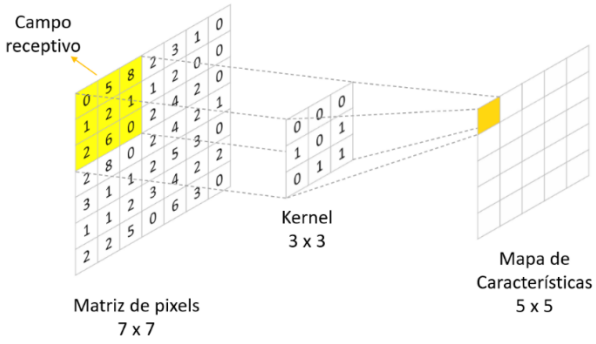
Os processos de extração e mapeamento das características são realizados nas camadas convolutivas, em que cada neurônio será conectado a um conjunto de pixels da imagem e a cada uma destas conexões serão atribuídos conjuntos de pesos a partir de uma matriz denominada de filtro de convolução, mas também conhecida como *kernel* (VARGAS, PAES e VASCONCELOS, 2016). O filtro de convolução será responsável pela detecção e extração de características de uma imagem e, posteriormente, o conjunto de saída resultante deste processo será armazenado na matriz denominada de mapa de características (*features map*) (LECUN *et al.*, 1998).

As dimensões do filtro de convolução irão depender do contexto para que estão sendo utilizados. Porém, para garantir uma detecção eficaz de bordas e outras características, o recomendável é que este filtro não possua dimensões menores que uma matriz (3×3). O tamanho do filtro definirá o tamanho da vizinhança na matriz de *pixel* da imagem de entrada, em que ocorrerá o processo. Portanto, é importante também definir o passo (*stride*) que definirá quantos *pixels* serão pulados entre cada janela do campo receptivo (VARGAS, PAES e VASCONCELOS, 2016).

É nesta camada que ocorre o processo que garante a robustez da CNN em relação a variações na imagem, tendo em vista que a partir de um certa característica seja detectada a sua posição real não é mais importante, apenas sua posição em relação a outras características detectadas. Se uma imagem for alterada, de forma que seus *pixels* forem deslocados, o mapa de características terá suas unidades de saída também deslocadas na mesma direção, mantendo a forma inalterada (LECUN *et al.*, 1998).

Portanto, se consideramos um exemplo onde uma imagem de entrada possua uma matriz de *pixels* de dimensões (7×7) e que se aplicarmos um *kernel* de dimensões (3×3) para realizar a detecção das características da imagem, utilizando um *stride* igual a 1, iremos obter um mapa de características de dimensões (5×5) (Figura 11). Isso se dá, pois com a utilização de um passo igual a 1, o campo receptivo irá percorrer os pixels da imagem pulando sempre um pixel, preenchendo o mapa de característica sempre linha por linha.

Figura 11 – Representação da operação de convolução aplicado em uma matriz de pixels de uma imagem.



Fonte: Elaborado pelo autor, 2023.

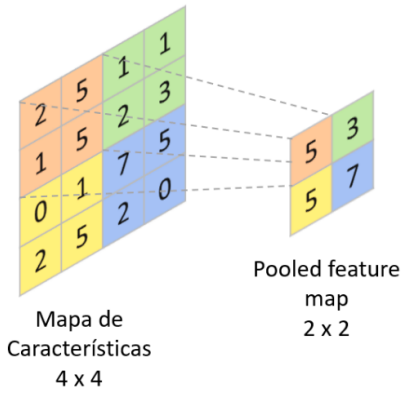
2.4.2 Camadas de Subamostragem

Geralmente, após cada camada convolutiva, é adicionado uma camada de subamostragem que realizará um processo de agrupamento, ou também conhecido como *pooling*. O *pooling* reduzirá a dimensão espacial do mapa de características de forma que a quantidade de parâmetros e a complexidade da rede diminua, e assim, o processo de treinamento se torna mais rápido, porém, mantendo a invariância espacial (VARGAS, PAES e VASCONCELOS, 2016; ALOYSIUS e GEETHA, 2017).

Algumas das operações de *pooling* existentes é o *max pooling*, processo de subamostragem com base no máximo local, o *min pooling*, processo realizado com base no mínimo local e ainda temos o *average* pooling, processo de subamostragem utilizará o valor médio local. Ainda assim, Aloysius e Geetha (2017) citam outros como o *stochastic pooling*, *spectral pooling*, *spatial pyramid pooling* e *pooling de ordem múltipla*.

Contudo, a operação mais utilizada e conhecida é o *max pooling*, onde se considera um mapa de características com dimensões (4×4) e aplica-se um *max pooling* de dimensões (2×2) com um *stride* igual a 2, obtendo uma matriz denominada de *pooled feature map* de dimensões (2×2), como pode ser visualizado na Figura 12.

Figura 12 - Representação da aplicação do *max pooling* em um mapa de características, com um *stride* igual a 2 e separada por cores a aplicação de cada *pooling*.



Fonte: Elaborado pelo autor, 2023.

2.4.3 Flattening e a MLP

Para realizar a conexão da parte convolucional da CNN com a camada totalmente conectada onde utilizaremos uma MLP, é necessário utilizar alguma operação que realize um processo de transformação unidimensional. Portanto, após esse processo, não pode existir mais nenhuma camada convolutiva, tendo em vista que esta etapa convolucional resulta em dados organizados espacialmente (ALOYSIUS e GEETHA, 2017).

Para realizar este procedimento, existe a operação descrita como *flattening* em que, por exemplo, uma matriz (2×2) resultará em um vetor com 4 valores. Porém, existem outras abordagens para este processo, como o *global average pooling* (GAP), que calcula uma média global em cada sinal de ativação da última camada convolucional, gerando um vetor de uma dimensão (ZHOU, KHOSLA, *et al.*, 2016).

A utilização do GAP pode trazer vantagens significativas no processo de conexão entre a parte convolucional da rede e a parte do módulo de classificação, pois, a sua aplicação diminui a propensão de *overfitting* nesta camada por não possuir parâmetros treináveis, pois trata-se de uma estrutura mais nativa a rede de convolução do que a opção do *flatten layer* (LIN, CHEN e YAN, 2013).

2.5 Transfer Learning

A transferência de aprendizagem (*transfer learning*), é uma técnica utilizada para transferir o conhecimento já aprendido por um modelo de rede neural base. Este modelo é geralmente treinado em base dados gigantescas de proposto geral como a base de imagens do

ImageNet, assim, reaproveitando ou transferindo o conhecimento adquirido para uma segunda rede neural de destino para a realização de uma determinada tarefa (YOSINSKI *et al*., 2014; VOGADO *et al*., 2019). Essa transferência de conhecimento se dará justamente pela transferência dos pesos pré-treinados na rede neural base para a de destino.

De acordo com Yosinski *et al.* (2014), o processo mais comum na utilização desta técnica é, após o treinamento de uma rede neural base, transferindo as primeiras $\boldsymbol{n}$ camadas para as primeiras $\boldsymbol{n}$ camadas de uma rede de destino, em que as camadas restantes serão treinadas para realizar alguma tarefa em específico. Porém, existe a possibilidade de realizar um treinamento nessas primeiras camadas que dependerá da quantidade dados ou da quantidade de parâmetros existentes nela, além da disponibilidade de um certo poder de processamento da máquina.

Essa técnica é bastante eficiente quando precisamos aproveitar a capacidade máxima de uma CNN, pois precisa ser treinada em uma massiva base dados para treinar os milhões de parâmetros que a constituem, resultando em uma necessidade de poder computacional mais elevado (VOGADO, *et al*., 2019).

Quanto a sua aplicação, pode-se dividir em dois processos distintos. O primeiro seria a transferência total da CNN pré-treinada, ou seja, além de utilizarmos seu módulo extrator de características também utiliza-se suas camadas totalmente conectadas, o módulo de classificação. Já o segundo processo se dá quando realiza-se apenas a transferência do módulo extrator da CNN pré-treinada, ou seja, apenas da parte convolucional e substituindo as camadas totalmente conectadas por outras camadas criadas para a realização da tarefa em questão (TAJBAKHSH, *et al*., 2016).

Além da transferência dos pesos de uma rede base para uma de destino, pode ser realizado diferentes tipos de ajustes, chamado de ajuste fino, do inglês *fine-tuning*. Existem dois tipos de ajustes finos que podem ser realizados. O primeiro é chamado de *shallow fine-tuned* (ajuste fino superficial), em que apenas as últimas da CNN serão retreinadas, mantendo congeladas as camadas anteriores com o objetivo de manter o conhecimento adquirido no pré-treinamento da rede base. O segundo ajuste é chamado de *deeply fine-tuned* (ajuste fino profundo) em que será realizado o treinamento total da rede, tanto a parte convolucional quanto a parte de classificação (IZADYYAZDANABADI *et al*., 2018; VOGADO *et al*., 2019).

Portanto, o uso do *shallow fine-tuned* visa treinar apenas as camadas que são consideradas mais especificas, deixando os pesos, das camadas anteriores a que será aplicado o ajuste fino, intactos o que beneficia no tempo de treinamento mais reduzido e menor requisição de poder computacional. Já a aplicação do *deeply fine-tuned* tende a resultar em uma

maior eficácia, porém, requer um poder de processamento mais elevado e um maior tempo de treinamento, e ainda, necessita de uma quantidade massiva de dados para treinar todos os milhões de parâmetros existentes (VOGADO *et al*., 2019).

2.6 Data Augmentation

Um banco de dados bastante desbalanceado pode aumentar os riscos de sobre ajuste no modelo, denominado *overfitting*. No *overfitting*, o modelo se adequa tanto aos dados de treinamento que perde sua capacidade de generalizar para novos dados. Na área de classificação de imagens por meio de modelo *deep learning*, é bastante comum bases de treino em quantidades bastantes desbalanceada, com isso se faz necessário utiliza alguma técnica que possibilite minimizar este problema.

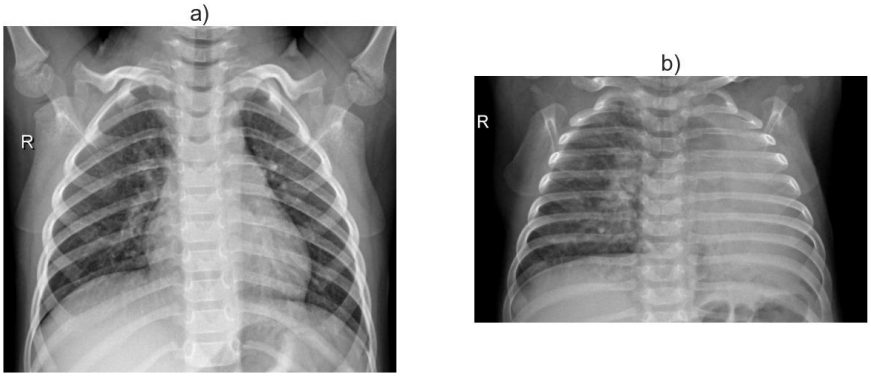
O uso do *data augmentation* (aumento de dados) é um processo que visa diminuir os problemas de sobreajustes, inflando artificialmente a base de dados com novas imagens criadas a partir de processos como o *data warping* (deformação dos dados). No *data augmentation* serão geradas novas imagens utilizando técnicas de mudança de cor, rotação, recortes e dentre outros tipos de transformações que podem ser aplicadas nas imagens existentes. Um outro processo é o *oversampling* (superamostragem) que criam instâncias sintéticas e as adicionam na base de treino (SHORTEN e KHOSHGOFTAAR, 2019).

3 METODOLOGIA

3.1 Base de imagens

O banco de imagens é composto por radiografias pediátricas de crianças com idades de um a cinco anos (Figura 13). Foram coletadas e disponibilizadas por Kermany, Goldbaum, *et al*. (2018) sendo de pacientes do *Guangzhou Women and Children's Medical Center*, localizado na cidade de Guangzhou na China, composta por radiografias normais e com a presença de pneumonia bacteriana e viral.

Figura 13 – Radiografias pediátricas do banco de dados de uma criança com o pulmão normal (a) e de uma criança com a presença de pneumonia em seus pulmões (b).



Fonte: Kermany, Goldbaum, *et al*. (2018).

Originalmente, esta base de imagens, é formado por um total de 5856 radiografias divididas em duas categorias: Normal e Pneumonia. Para a base de treinamento tem-se para categoria Pneumonia uma quantidade de 3875 radiografias, sendo 2530 de pneumonia bacteriana e as outras 1345 de pneumonias virais. Para a categoria Normal, tem-se apenas 1341 imagens. Já para a base de teste foram utilizados um total de 624 imagens, sendo que 234 correspondem a classe Normal, e as demais 390 são da classe Pneumonia com 242 imagens para as bacterianas e 148 para as virais. Ainda tem-se a base de validação com 8 imagens de radiografias para cada uma das categorias.

A divisão das imagens para os grupos de treinamento, teste e validação foram realizados pelos próprios autores e neste trabalho todas as radiografias da classe Pneumonia serão consideradas únicas e não separadas como virais ou bacterianas, sendo assim, iremos realizar a classificação para identificar se a criança possui ou não tal doença pulmonar.

3.2 Balanceamento da Base de Treinamento

A base de imagens de treinamento possui um desequilíbrio significativo entre as duas categorias, em que, 25,71% do total de imagens de treinamento é da classe Normal, ou seja, uma diferença de 65,39% em relação ao total de imagens da classe Pneumonia.

Sendo assim, foi realizado um processo de *oversampling* para o aumento da quantidade de imagens da classe minoritária, com a finalidade de equalizar a quantidade para ambas as duas classes. Neste caso, será utilizado o método do *data augmentation* através da biblioteca *Albumentations* (BUSLAEV *et al*., 2020) no *python*, onde as novas imagens foram geradas utilizando a métodos de redimensionamentos, recortes, mudanças de brilho e contrastes de forma aleatória utilizando a função *Compose*.

Para a aplicação, foi usado uma amostra de 845 radiografias do banco de imagens de treinamento da classe Normal, que correspondem a um total de 1341, ou seja, cerca de 63,01% das imagens originais para gerar novas imagens a partir da aplicação do método do *data augmentation*. De cada imagem amostrada foi criado mais 3 imagens, totalizando uma quantidade de 3876 imagens, onde foi selecionada aleatoriamente uma imagem das que foram geradas para ser excluída e igualar a quantidade em ambas as categorias na base de treinamento.

3.3 Arquiteturas Convolucionais

Neste trabalho, através da aplicação do método do *transfer learning*, foram utilizadas duas arquiteturas convolucionais distintas, porém, bastantes conhecidas e utilizadas em diversas aplicações sendo treinadas utilizando o banco de dados de imagens do ImageNet, aplicando o ajuste fino *shallow fine*-tuned, em que, apenas 25% das últimas camadas convolucionais de cada modelo foram retreinadas.

Ao importarmos uma CNN, pode-se optar por utilizar a sua rede neural densa que foi aplicada durante seu processo de treinamento. Contudo, foram utilizadas apenas as suas camadas convolucionais, ou seja, apenas o seu módulo extrator de características em que será conectado a uma RNA criada com as mesmas camadas para as duas, diferindo apenas na quantidade de neurônios artificiais utilizados.

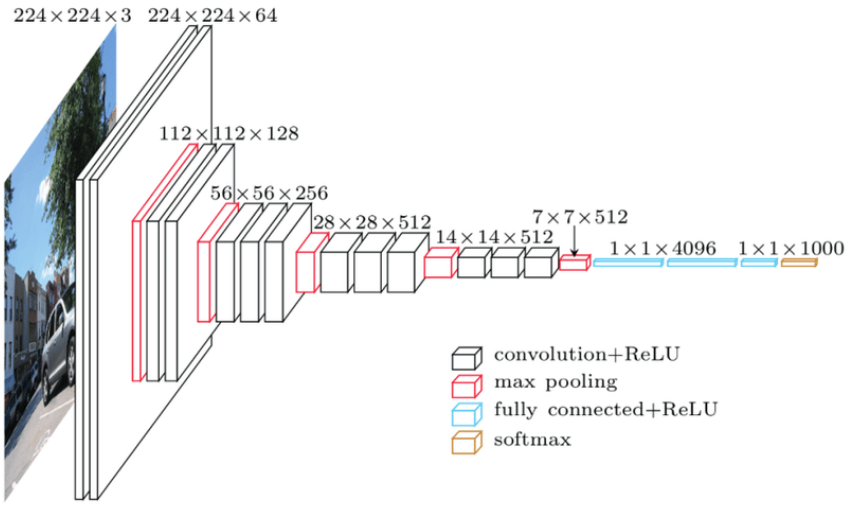
3.3.1 VGG-16

Um dos principais modelos de CNN já criados e se destacando justamente por possuir uma arquitetura mais simples, porém bastante eficaz e com uma quantidade de parâmetros bastante alta, a VGG-16 é composto por 23 camadas contando com as 3 camadas totalmente conectadas (Módulo de classificação) em que, 16 destas camadas possuem pesos treináveis, daí a nomeação da arquitetura (SIMONYAN e ZISSERMAN, 2014). Este modelo se utiliza de um filtro bastante pequeno de dimensões (3 x 3) em sua camada de convolução seguidos, a partir da camada 10, de um outro de dimensões (1 x 1) para realizar uma transformação linear nos canais de entrada (Figura 14) (SIMONYAN e ZISSERMAN, 2014).

A proporção de pixels que uma imagem deve ter para ser introduzida neste modelo é de (224 x 224) em formato RGB, já as camadas de subamostragem são realizadas utilizando o *max pooling* de dimensões (2 x 2) pixels (SIMONYAN e ZISSERMAN, 2014). Podemos analisar, a partir da Figura 14, a representação da VGG-16 junto com seu módulo de classificação.

Então, serão utilizadas apenas as camadas correspondentes ao processo convolucional que são as 19 primeiras camadas, contando com a camada de entrada, em que, apenas 13 possuíram pesos a serem treinados (Figura 14).

Figura 14 – Estrutura da Rede Neural Convolucional VGG-16.



Fonte: VGG16 - Convolutional Network for Classification and Detection (2018).

3.3.2 Inception-ResNet-V2

A Inception-ResNet-v2 é uma versão híbrida entre a arquitetura *Inception*, que tende a ser bastante profunda e densa, substituindo suas camadas de concatenação de filtros por meio

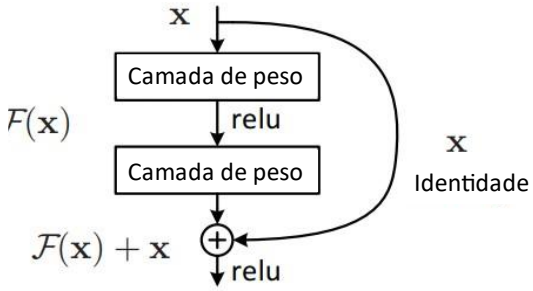
de conexões residuais advindas de arquiteturas de Redes Residuais, também conhecidas como *ResNet* (SZEGEDY, IOFFE, *et al.*, 2017). A aplicação destas conexões residuais visa a tornar o processo de treinamento mais eficiente à medida que a profundidade da rede neural aumenta diminuindo o processo de degradação da rede neural (HE, ZHANG, *et al.*, 2016).

Podemos definir este processo, bem como definem He, Zhang *et al*. (2016), que ao invés de empilhar várias camadas em blocos convolucionais, o método seria aplicado de forma que a saída destes blocos resultem na soma do mapeamento residual $\boldsymbol{H(x)}$ pelo valor da entrada $\boldsymbol{x}$ no bloco, de forma que resulte em uma aproximação de uma função residual $\boldsymbol{F(x)}$**:**

$$\boldsymbol{F(x) = H(x) + x} \tag{3.1}$$

em que, ainda podemos reescrever esta fórmula para apenas $\boldsymbol{y = F(x) + x}$ (Figura 15).

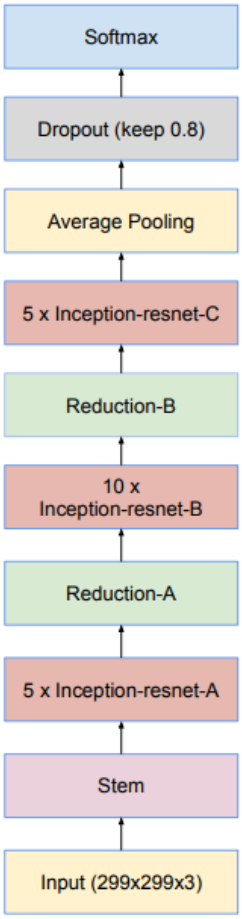
Figura 15 - Construção de um bloco de aprendizado residual.



Fonte: He, Zhang, *et al.* (2016)

A imagem que deverá ser introduzida a partir da camada de entrada deve possuir uma proporção de (299 x 299) *pixels* e em RGB. Assim como para a arquitetura da VGG-16, os pixels foram padronizados dividindo todos por 255. Como já mencionado, a Inception-ResNet-V2 é composta por blocos de convolução com a aplicação do método para o cálculo residual, como pode ser representado na Figura 16 nos blocos de *inception-resnet* A, B e o C.

Figura 16 – Estrutura da rede neural convolucional Inception-ResNet-V2.



Fonte: He, Zhang, *et al.* (2016)

3.3.3 RNA criada

Para o módulo de classificação foram criados duas RNA totalmente conectadas, uma para o VGG-16 e outra para a Inception-ResNet-V2, em que o processo de conexão com o módulo extrator será pelo método GAP. Tem-se ainda duas camadas escondidas utilizando a função de ativação *ReLU*. A quantidade de neurônios artificias utilizados nestas camadas será dado pela seguinte fórmula 3.2.

$$\frac{\boldsymbol{QN + 2}}{\boldsymbol{2}} \tag{3.2}$$

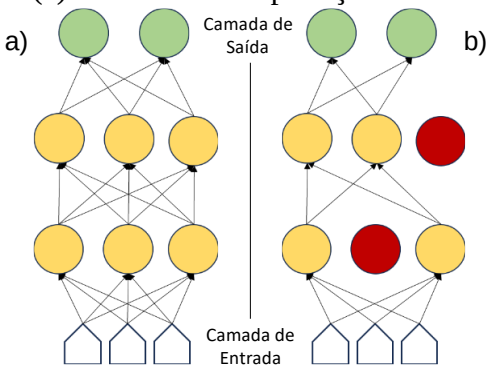
em que $\boldsymbol{QN}$ é a quantidade de neurônios da última camada da parte convolucional da arquitetura.

Para a VGG-16 a última camada convolucional terá 512 neurônios, portanto, para as camadas escondidas iremos utilizar uma quantidade de 257 unidades de processamento.

Enquanto para a Inception-ResNet-V2 em sua última camada convolucional existem 1536 neurônios, com isso utilizaremos 769 neurônios nas duas camadas escondidas da RNA (Tabela 2).

Cada uma das camadas escondidas são seguidas por uma camada de *dropout* com uma taxa de 20%, em que alguns dos nodos tiveram suas ativações zeradas fazendo com que ficassem inativos durante o processo de treinamento (ALOYSIUS e GEETHA, 2017). Neste caso, as informações de 20% dos nodos serão descartados. Um exemplo da aplicação deste método pode ser visto na Figura 17b onde os neurônios em vermelhos foram inativados. Para a última camada onde será enfim realizado o processo de classificação, foi utilizado apenas 2 neurônios com a função de ativação *softmax*.

Figura 17 – Ilustração da aplicação do método *dropout* comparando uma RNA sem *dropout* (a) e outra com a aplicação do método (b).



Fonte: Elaborado pelo próprio autor, 2023.

Ao combinarmos apenas a parte de ambas as arquiteturas com a RNA criada a quantidade de parâmetros diferem, pois existe uma quantidade de neurônios específico a ser utilizado para cada uma das arquiteturas. Esta informação pode ser vista na Tabela 2, onde notamos que a quantidade de parâmetros finais na RNA criada foi menor justamente quando a quantidade de neurônios foi menor, no caso da VGG-16.

Tabela 2 – Esquema da quantidade de neurônio e parâmetros das camadas da RNA criada para o módulo de classificação incorporado tanto no modelo com a Inception-ResNet-V2 e com a VGG-16.

Camadas	Inception-ResNet-V2		VGG-16	
	Neurônios	Parâmetros	Neurônios	Parâmetros
global_av_pooling2d	1536	0	512	0
dense	769	1.181.953	257	131.841
dropout	769	0	257	0
dense_1	769	592.130	257	66.306
dropout	769	0	257	0
dense_2	2	1.540	2	516
Total	**4614**	**1.775.623**	**1542**	**198.663**

Fonte: Elaborado pelo autor, 2023.

Na Tabela 3, é apresentada uma contagem do número de camadas e parâmetros presentes em cada uma das CNN's utilizadas, tanto para a sua arquitetura original onde tem-se uma rede completamente conectada responsável pelo processo de classificação da própria arquitetura, quanto para a junção do seu módulo extrator com uma RNA criada especificamente para o problema em questão.

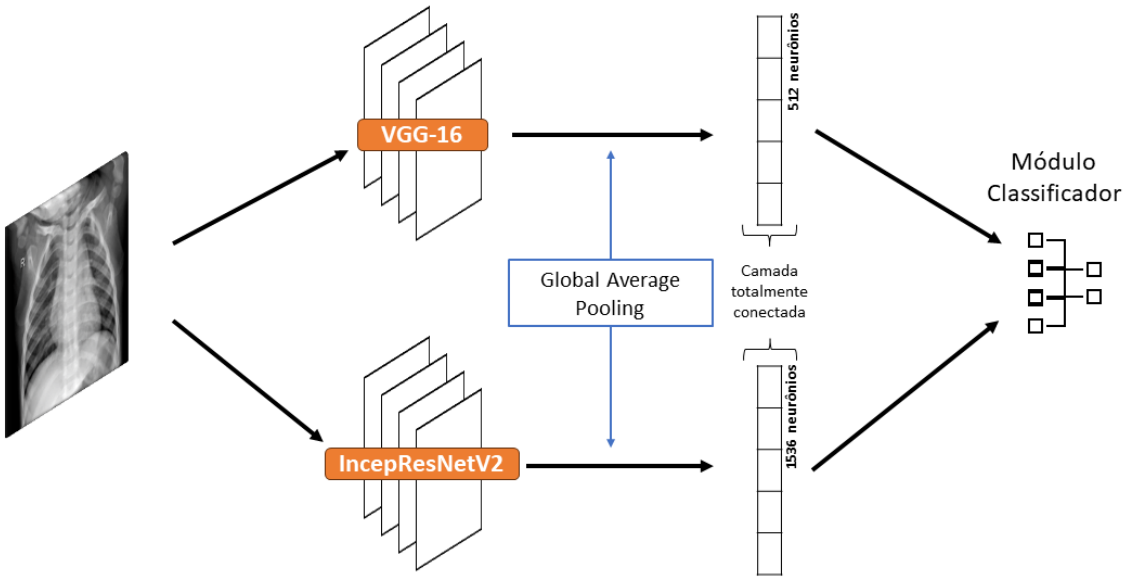
No caso da arquitetura da VGG-16, ao utilizarmos apenas o módulo extrator e combiná-lo com a RNA criada, a quantidade de parâmetros cai significativamente em relação a mesma arquitetura usando o próprio módulo de classificação. Isso se dá pelo fato de que a rede completamente conectada, utilizada na VGG-16, possui uma quantidade de parâmetros alta, o que aumenta o tempo de treinamento. De forma resumida pode-se esquematizar este processo através da figura 18.

Tabela 3 – Quantidades de camadas e parâmetros (em milhões) para cada CNN utilizadas.

CNN	Original		Mod. Extract + RNA	
	Camadas	Parâmetros	Camadas	Parâmetros
Inception-ResNet-V2	782	55M	786	56M
VGG-16	23	38M	25	14M

Fonte: Elaborado pelo autor, 2023.

Figura 18 – Esquema da implementação das CNN utilizadas com o módulo de classificação, a RNA criada.



Fonte: Elaborado pelo próprio autor, 2023.

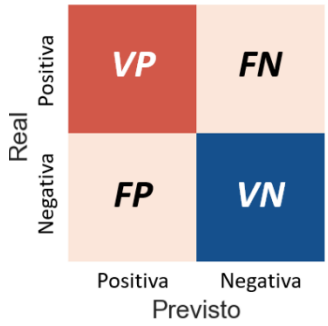
3.4 Métricas de avaliação e comparação entre os modelos

3.4.1 Matriz de Confusão

Para a avaliação do desempenho de um modelo de aprendizagem com foco na classificação, incialmente, foi construída uma matriz de confusão para cada um dos modelos treinados. Essa aplicação resulta em uma representação tabular em que cada linha desta matriz bidimensional representa as classes reais, enquanto, as colunas representa as classes previstas pelo modelo (GÉRON, 2019).

Sendo assim, a matriz de confusão retorna quatro resultados possíveis: Verdadeiros Positivos (VP), quando uma classe realmente positiva for prevista corretamente como positiva, Verdadeiros Negativos (VN), instâncias que foram classificadas como negativas sendo que eram realmente da classe negativa, e tem-se ainda os Falso Negativos (FN) e os Falsos Positivos (FP), respectivamente, em casos onde a classe real era positiva, porém foi classificada como negativa e quando a classe real for negativa, porém foi classificada como positiva (Figura 19) (FAWCETT, 2006).

Figura 19 – Representação da matriz de confusão.



Fonte: Elaborado pelo próprio autor, 2023.

3.4.2 Acurácia

Neste trabalho, a classe Positiva (VP) será representado por Pneumonia e a Negativa (VN) pela classe Normal. Com base nos valores calculados e representados pela matriz de confusão, pode-se calcular as métricas para a avaliação dos modelos como a acurácia que medirá a porcentagem de acertos totais do modelo, dada pela razão entre o total de acertos e o total de instâncias classificadas, sendo definida na equação 3.3.

$$\boldsymbol{acurácia} = \frac{\boldsymbol{VP} + \boldsymbol{VN}}{\boldsymbol{VP} + \boldsymbol{FN} + \boldsymbol{VN} + \boldsymbol{FP}} \tag{3.3}$$

Porém, em casos com bases de treino desbalanceadas, a utilização da acurácia como única medida de avaliação não é recomendada, se fazendo necessário o cálculo e utilização de outras métricas para a avaliação do desempenho do modelo.

3.4.3 Precisão

Com isso, outra métrica utilizado foi a precisão que avalia a quantidade de valores positivos classificados pelo modelo em relação a todas as previsões positivas. Ou ainda, pode-se definir como sendo uma acurácia referente a previsões positivas do modelo (GÉRON, 2019), dado pela equação 3.4.

$$\boldsymbol{Precisão}_{(\boldsymbol{Pneumonia})} = \frac{\boldsymbol{VP}}{\boldsymbol{VP} + \boldsymbol{FP}} \tag{3.4}$$

Além de calcularmos a precisão para a classe positiva, também foi calculado para a classe negativa (classe Normal), definida na equação 3.5.

$$\boldsymbol{Precisão}_{(\boldsymbol{Normal})} = \frac{\boldsymbol{VN}}{\boldsymbol{VN} + \boldsymbol{FN}} \tag{3.5}$$

3.4.4 Sensibilidade e Especificidade

De forma conjunta com a avaliação por parte da precisão do modelo, foi utilizado também a sensibilidade (*recall)* que mediu a capacidade do classificador em conseguir prever de forma correta todos os casos que eram positivos (GÉRON, 2019), em que:

$$\boldsymbol{Sensibilidade} = \frac{\boldsymbol{VP}}{\boldsymbol{VP} + \boldsymbol{FN}} \tag{3.6}$$

Ainda foi calculado a especificidade para medirmos a Taxa de Verdadeiros Negativos (TVN), de maneira análoga a definição para a classe positiva, foi medida a proporção de instâncias negativas prevista pelo modelo que realmente eram negativas, dada na equação 3.7.

$$\boldsymbol{Especificidade} = \frac{\boldsymbol{VN}}{\boldsymbol{VN} + \boldsymbol{FP}} \tag{3.7}$$

3.4.5 F1-score

Para encontrar um modelo equilibrado em termos de precisão e sensibilidade, ou especificidade, foi utilizado o F1-score que nada mais é que uma média harmônica entre essas métricas. Portanto, foram calculados dois F1-score para cada modelo, um com base na classe Pneumonia e outro com base na classe Normal, utilizando o valor da sensibilidade e especificidade, como demonstrado nas equações (3.8) e (3.9).

$$\boldsymbol{F1 - score_{(Pneumonia)}} = \mathbf{2} \times \frac{\boldsymbol{Precisão} \times \boldsymbol{Sensibilidade}}{\boldsymbol{Precisão} + \boldsymbol{Sensibilidade}} \tag{3.8}$$

e

$$\boldsymbol{F1 - score_{(Normal)}} = \mathbf{2} \times \frac{\boldsymbol{Precisão} \times \boldsymbol{Especificidade}}{\boldsymbol{Precisão} + \boldsymbol{Especificidade}} \tag{3.9}$$

A utilização da média harmônica neste caso irá penalizar os classificadores que possuírem tanto uma precisão ou sensibilidades baixas, onde se ambas as métricas forem altas o F1-score também será alto, portanto, esse método é bastante eficaz em determinar modelos equilibrados e contribuindo para uma melhor comparação entre dois modelos (GÉRON, 2019).

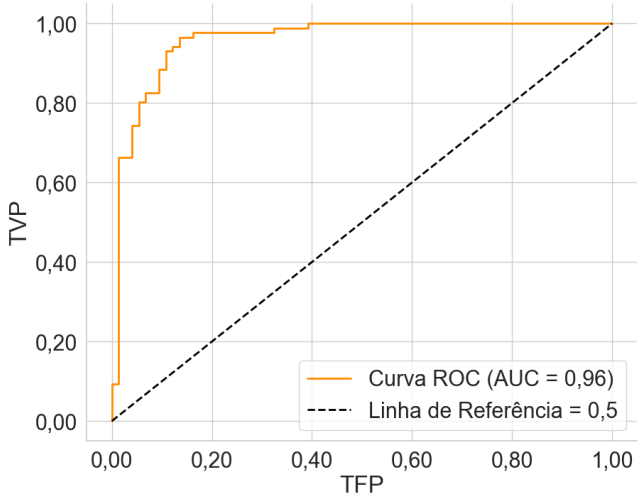
3.4.6 Curva ROC

Foi ainda utilizado a curva ROC (*Receiver Operating Characteristic*), definida por um gráfico um gráfico onde tem-se uma relação entre a Taxa de Verdadeiros Positivos (TVP), a sensibilidade do modelo, e a Taxa de Falsos Positivos (TFP) que pode ser calculada a partir da equação 3.10 onde calcula-se 1 menos a TVN, que é mesma taxa que denominada de especificidade.

$$\boldsymbol{TFP = 1 - Especificidade} \tag{3.10}$$

A partir do Gráfico 1, pode-se avaliar o modelo com base no valor da área abaixo da curva, ou como é denominada de AUC (*Area Under Curve*), em que quanto mais próximo o valor calculado do AUC for de 1, melhor é o modelo, todavia, se o AUC calculado se aproximar de 0,5, conclui-se que o modelo assume um comportamento totalmente aleatório durante o processo de classificação (GÉRON, 2019). A área com o AUC igual a 0,5 é delimitada no gráfico a partir de uma linha pontilhada que é chamada de linha de referência.

Gráfico 1 – Exemplo de uma curva ROC.



Fonte: Elaborado pelo próprio autor, 2023.

3.4.7 Curva PR

A curva PR (Precision - Recall) ou também chamada de curva Precisão – Revocação (Sensibilidade) tem sua utilização recomendada quando existe um desequilíbrio significativo na base de treinamento entre as classes. Este gráfico mostra uma relação entre a precisão do

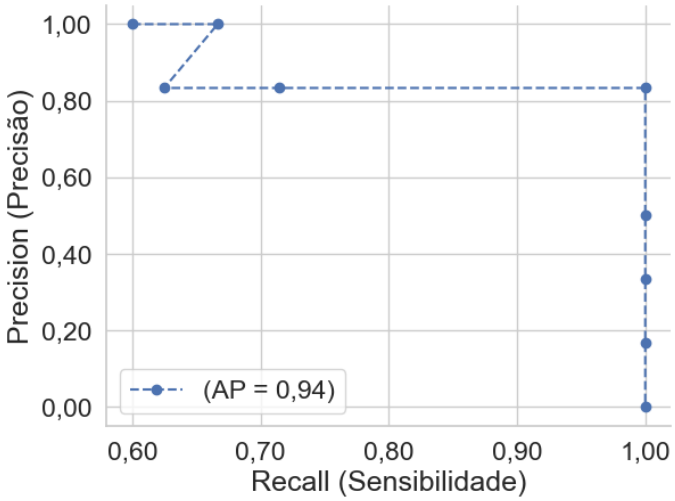
modelo e a sensibilidade, em que uma área significativa sob a curva indica alto desempenho, com uma alta precisão e uma alta sensibilidade (PEDREGOSA, VAROQUAUX, *et al.*, 2011).

Uma maneira de resumir a curva PR é por meio do cálculo da Precisão Média (*Average Precision* - AP), a qual consiste em uma média ponderada das precisões em vários pontos de decisão, utilizando como peso o aumento de recall em relação ao ponto anterior. Isso proporciona uma abordagem equilibrada para avaliar a capacidade do modelo de classificação em produzir resultados corretos e identificar com precisão a maioria das instâncias positivas.

$$\boldsymbol{AP} = \sum_{\boldsymbol{n}} (\boldsymbol{R_n} - \boldsymbol{R_{n-1}}) \times \boldsymbol{P_n} \tag{3.11}$$

Na equação (3.11) tem-se a fórmula para o cálculo do AP, onde realiza-se um somatório sobre cada ponto da curva PR. Onde $\boldsymbol{R_n}$ será o recall no ponto $\boldsymbol{n}$, $\boldsymbol{R_{n-1}}$ será o recall no ponto anterior ao ponto $\boldsymbol{n}$ e o $\boldsymbol{P_n}$ é a precisão calculado no ponto $\boldsymbol{n}$.

Gráfico 2 – Exemplo de uma curva PR.



Fonte: Elaborado pelo próprio autor, 2023.

3.5 Ferramentas utilizadas

Todos as análises e processos utilizados neste trabalho foram desenvolvidos utilizando a linguagem de programação *Python*, na versão 3.10.1, utilizando as bibliotecas *Seaborn* (WASKOM, 2021) e *Matplotlib* (HUNTER, 2007) para a criação dos gráficos e plotagem das imagens de raio-x. Além destes, foram utilizados o *OpenCV* (BRADSKI, 2000) para a leitura e manipulação das imagens, assim como o *Numpy* (HARRIS, MILLMAN, *et al.*, 2020), o *Pandas*

(MCKINNEY, 2010) e também a biblioteca *Albumentations* (BUSLAEV, IGLOVIKOV, *et al.*, 2020), e a biblioteca do *scikit-learn* (PEDREGOSA, VAROQUAUX, *et al.*, 2011).

Para a aplicação do método do *transfer learning* foi utilizado o *Keras* (CHOLLET, *et al.*,2015) através da biblioteca *Tensorflow* (ABADI, AGARWAL, *et al.*). Todas essas aplicações foram realizadas em um computador de uso pessoal, utilizando o processador Intel i5-10400, 6 núcleos físicos e 12 *threads* com uma frequência base de 2.90 GHz, 16 GB de memória Ram e uma Geforce GTX 1660 SC Ultra Gaming da EVGA com uma memória de 6GB GDDR5 e 1405 CUDA Cores.

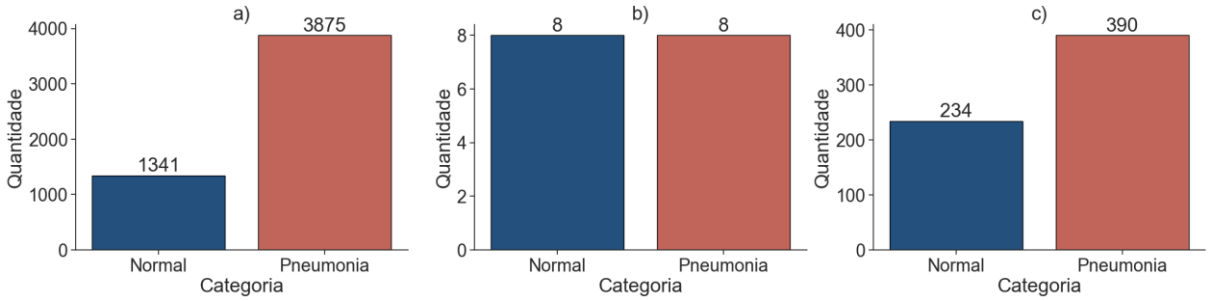
4 RESULTADOS E DISCUSSÕES

Nesta etapa, iremos analisar e comparar o desempenho dos modelos que utilizam a VGG-16 e a Inception-ResNet-V2 utilizando o banco de dados originais, onde tem-se presente um desbalanceamento entre as categorias Pneumonia e Normal, e a base de dados balanceada utilizando o processo de *data augmentation*. Com isso, após a realização do processo de treinamento para ambos os modelos em apenas 10 épocas, foi possível obter resultados satisfatórios.

4.1 Avaliação dos modelos

O banco de dados com as imagens originais possui um desbalanceamento significativo na base de treinamento, onde de um total de 5216 imagens de radiografias a categoria Normal possui 1341 e Pneumonia possui 3875 imagens, sendo uma diferença de 2534 radiografias entre as duas categorias (Gráfico 3a). Já na base de validação, utilizado durante a etapa de treinamento, é composta por 8 imagens de radiografias para cada categoria, como podemos verificar no Gráfico 3b. No Gráfico 3c tem-se a base de testes com 390 imagens para a categoria Pneumonia e 234 para a classe Normal.

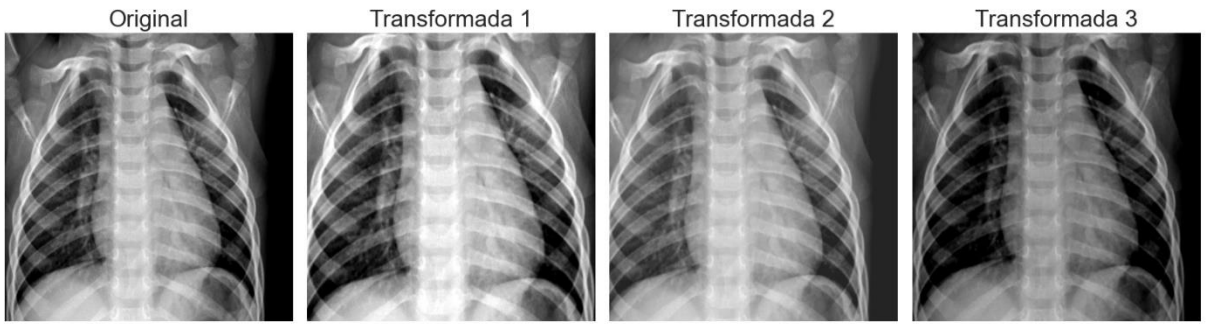
Gráfico 3 - Quantidade de imagens de radiografias nas bases de treinamento (a), de validação (b) e de teste (c).



Fonte: Elaborado pelo autor, 2023.

Realizando o processo do *data augmentation* para a criação de novas imagens com a finalidade balancear a base de treinamento, foi possível equalizar ambas as classes. Um exemplo de como a aplicação desta técnica resultou nas novas imagens geradas pode ser vista na Figura 20, onde temos que para cada imagem original da classe Normal, foram geradas novas três imagens listadas como transformada 1, 2 e 3.

Figura 20 – Exemplo da criação das imagens aplicando as transformações pelo processo do *data augmentation*.



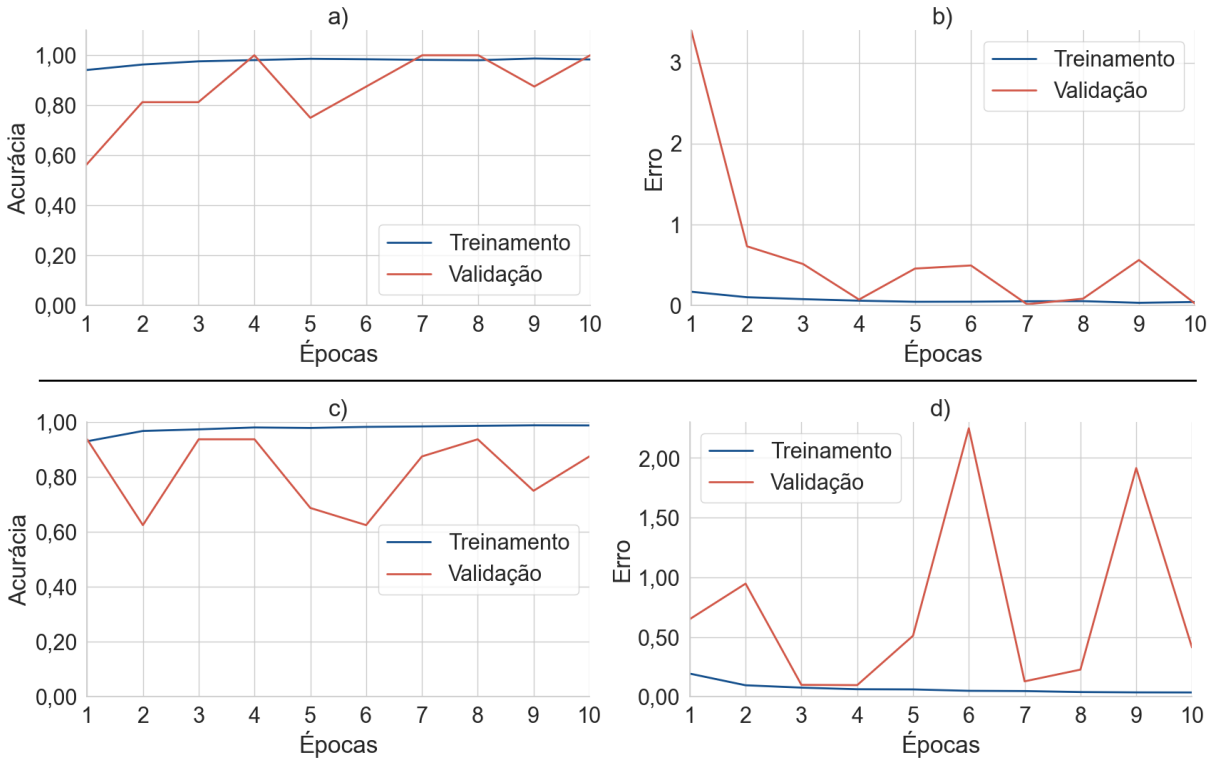
Fonte: Elaborado pelo autor, 2023.

Avaliando a evolução da acurácia no modelo da Inception-ResNet-V2 (Gráfico 4a) no processo de aprendizagem, podemos verificar que enquanto na base de treinamento o modelo apresentava uma leve constância no aumento do valor da acurácia ao longo das etapas de treinamento. Já na base de validação, apresentada no Gráfico 4b, ocorreram algumas flutuações significativas que podem indicar problemas de sobreajuste do modelo.

Partindo para a avaliação do modelo treinado na base balanceada, nota-se que o problema verificado acima acaba piorando ainda mais, o modelo aparenta possuir problemas em classificar imagens na base de validação, ressaltando o problema da falta de calibração que ele possui para essa tarefa. No Gráfico 4c, o modelo não apresentou um bom desempenho em relação a acurácia, com sucessivas quedas, principalmente entre as épocas 4 e 6, evidenciando que neste intervalo não houve um aprendizado significativo.

Em relação ao gráfico da evolução do erro (Gráfico 4d), o modelo apresenta um cenário bastante negativo, enquanto o cálculo do erro para a base de treinamento cai e se aproxima do zero, na base de validação parece demonstrar problemas que o modelo possui em generalizar para dados de fora da base de treinamento, diminuindo assim a eficácia no processo final de classificação.

Gráfico 4 – Curva de aprendizagem do modelo convolucional da Inception-ResNet-V2 em relação a evolução da acurácia ao longo das épocas de treinamento (a) para a base desbalanceada e (c) para a balanceada, e a evolução do erro durante as etapas de treinamento (b) usando a base desbalanceada e (d) para a balanceada.



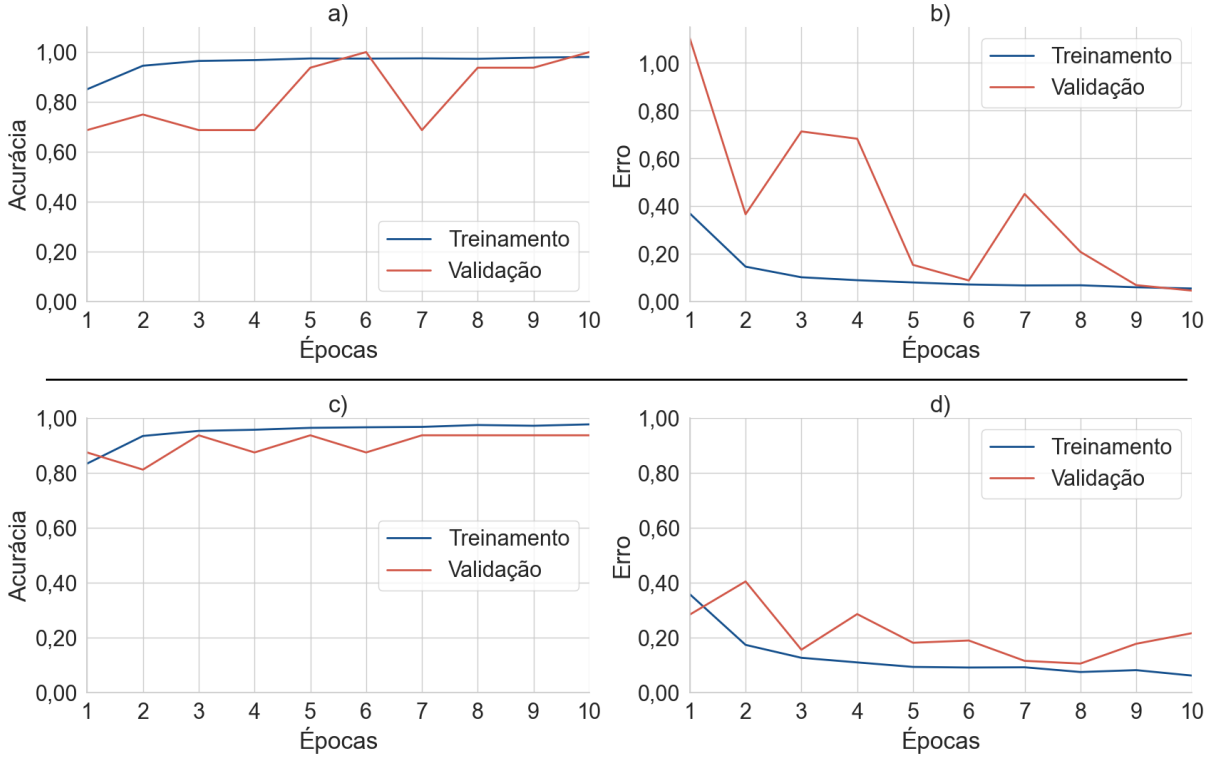
Fonte: Elaborado pelo autor, 2023.

Os mesmos problemas podem ser verificados avaliando o Gráfico 5a e 5b, referente ao processo de treinamento do modelo da VGG-16 na base desbalanceada, onde possuímos as mesmas evidências de que os modelos parecem estar se ajustando demais a base de treinamento. Contudo, quando o modelo foi treinado usando base balanceada ocorreu uma melhora significativa no poder de generalização do modelo na obtenção de resultados satisfatórios na base de validação.

A partir do Gráfico 5c o desempenho em relação a acurácia apresentou leves flutuações e uma estagnação no processo que se inicia na 7ª época, persistindo até o final do treinamento. Já quando observamos o Gráfico 5d é possível verificar uma tendência de queda do erro na base de validação até a 8ª época de treinamento que apresentou um leve aumento, porém, a evolução da taxa se mostrou mais suave e seguindo a taxa com base nos dados de treinamento.

Essa situação é um indício de que este modelo da VGG-16, treinado na base de dados que foi realizado o processo de *data augmentation* para o balanceamento entre as categorias a serem classificadas, obteve uma melhor calibração na etapa de aprendizagem.

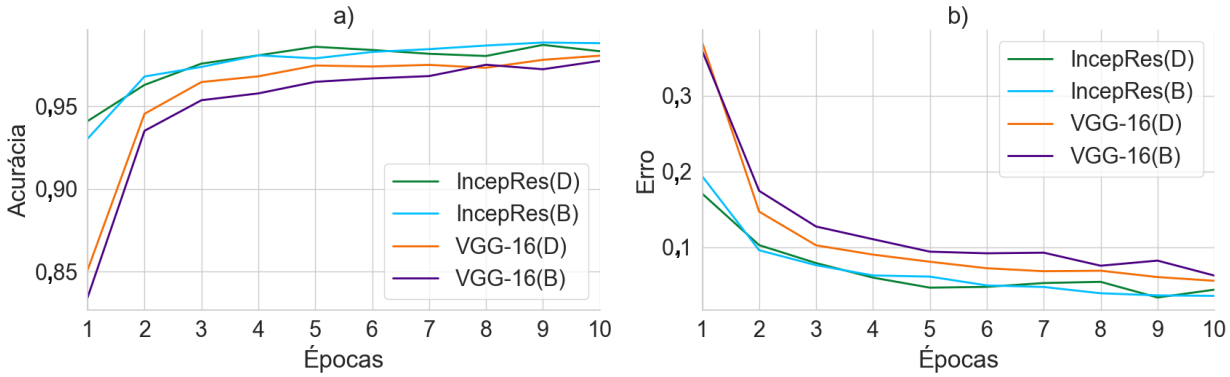
Gráfico 5 - Curva de aprendizagem do modelo convolucional da VGG-16 em relação a evolução da acurácia ao longo das épocas de treinamento (a) para a base desbalanceada e (c) para a balanceada, a evolução do erro durante as etapas de treinamento (b) usando a base desbalanceada e (d) para a balanceada.



Fonte: Elaborado pelo autor, 2023.

Com isso, realizando uma análise comparativa entre os dois modelos, conforme evidenciado no Gráfico 6, tanto no que diz respeito à progressão da acurácia na base de treinamento, quanto ao comportamento do erro, observa-se uma notável similaridade de desempenho. Esta observação é especialmente destacada quando ampliamos o eixo y para o intervalo de [0,1]. No entanto, é crucial ressaltar que, nesse contexto, o modelo convolucional utilizando a Inception-ResNet-V2 demonstra resultados superiores ao longo desse processo.

Gráfico 6 – Comparação entre os modelos convolucionais da Inception-ResNet-V2 e VGG-16 em relação a evolução da acurácia ao longo das épocas de treinamento (a) e a evolução do erro durante as etapas de treinamento (b) utilizando a base balanceada e desbalanceada.



***B**: Base Balanceada; **D**: Base Desbalanceada;*

Fonte: Elaborado pelo autor, 2023.

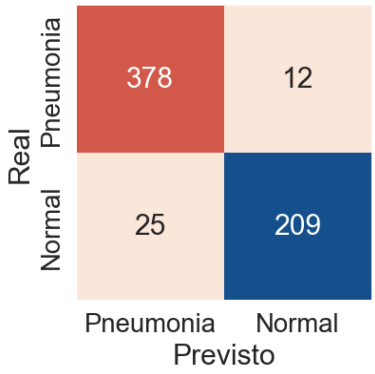
4.2 Etapa de classificação

Após o processo de treinamento e classificação foi construída a matriz de confusão para ambas as arquiteturas, com a finalidade de compreender melhor o desempenho dos dois modelos comparando as previsões realizadas em relação as categorias reais das imagens.

4.2.1 Base de dados desbalanceada

Na Figura 21, o primeiro modelo que utiliza a CNN Inception-ResNet-V2, das 390 imagens da classe Pneumonia, classificou de forma correta 378 imagens como sendo de crianças com a doença pulmonar enquanto 12 crianças que possuíam pneumonia, o modelo classificou como Normal. Por outro lado, das 234 imagens da classe Normal, foram classificadas corretamente 209 imagens e as outras 25 foram classificadas erroneamente como possuindo pneumonia (Figura 21).

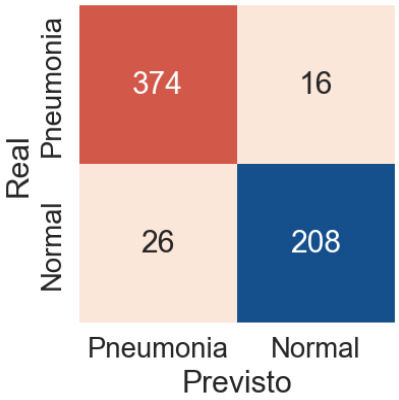
Figura 21 – Matriz de confusão para o modelo utilizando a Inception-ResNetV2.



Fonte: Elaborado pelo autor, 2023.

Já no modelo com a arquitetura VGG-16 (Figura 22), verificou-se que de todas as imagens que eram Pneumonia ele classificou de forma correta 374 imagens enquanto as 16 restantes foram classificadas como Normal. No entanto, para as imagens que eram da categoria Normal, o modelo classificou de forma correta 208 imagens, enquanto as restantes, total de 26, foram classificadas como possuindo pneumonia.

Figura 22 - Matriz de confusão para o modelo utilizando a VGG-16.



Fonte: Elaborado pelo autor, 2023.

Com isso, as informações necessárias advindas das matrizes de confusão acima, onde descreve o quantitativo dos VP, FP, VN e dos FN, pode-se calcular as métricas de avaliação de um modelo, como a precisão, sensibilidade, especificidade, F1-score e a acurácia para cada um dos modelos e realizar as devidas comparações. Como o problema é relacionado a área da saúde, precisamos sempre buscar uma menor quantidade possível nos FP e FN.

Ao analisar a precisão de ambos os modelos, notamos que o modelo da VGG-16 registrou uma acurácia de 93,26%, enquanto o modelo que utiliza a CNN Inception-ResNet-V2 alcançou uma acurácia ligeiramente superior, atingindo 94,07%. Contudo, é importante ressaltar que essa métrica pode não ser totalmente confiável, especialmente ao lidar com conjuntos de dados desbalanceados entre as classes a serem classificadas.

A partir da Tabela 4 e 5, observa-se que o modelo com a Inception-ResNet-V2 demonstrou resultados mais equilibrados em comparação com o modelo da VGG-16. Este último apresentou uma sensibilidade de 0,96 e uma especificidade de 0,89, valores semelhantes aos calculados para o modelo da Inception-ResNet-V2, que também obteve os mesmos valores para a sensibilidade e especificidade, porém, com a precisão da classe negativa sendo superior ao da VGG-16.

Esse equilíbrio na calibração dos modelos fica mais evidente no cálculo do F1-Score, em que, para as duas categorias a serem classificadas, ambos os modelos exibiram uma equidade notável no poder de detecção de pneumonia ou sua ausência nas radiografias pediátricas do conjunto de dados.

Tabela 4 – Resultados para o modelo da Inception-ResNet-V2.

Classes	Precisão	Sensibilidade	Especificidade	F1-Score
PNEUMONIA	0,94	0,97	-	0,95
NORMAL	0,95	-	0,89	0,92

Fonte: Elaborado pelo autor, 2023.

Tabela 5 - Resultados para o modelo da VGG-16.

Classes	Precisão	Sensibilidade	Especificidade	F1-Score
PNEUMONIA	0,94	0,96	-	0,95
NORMAL	0,83	-	0,89	0,91

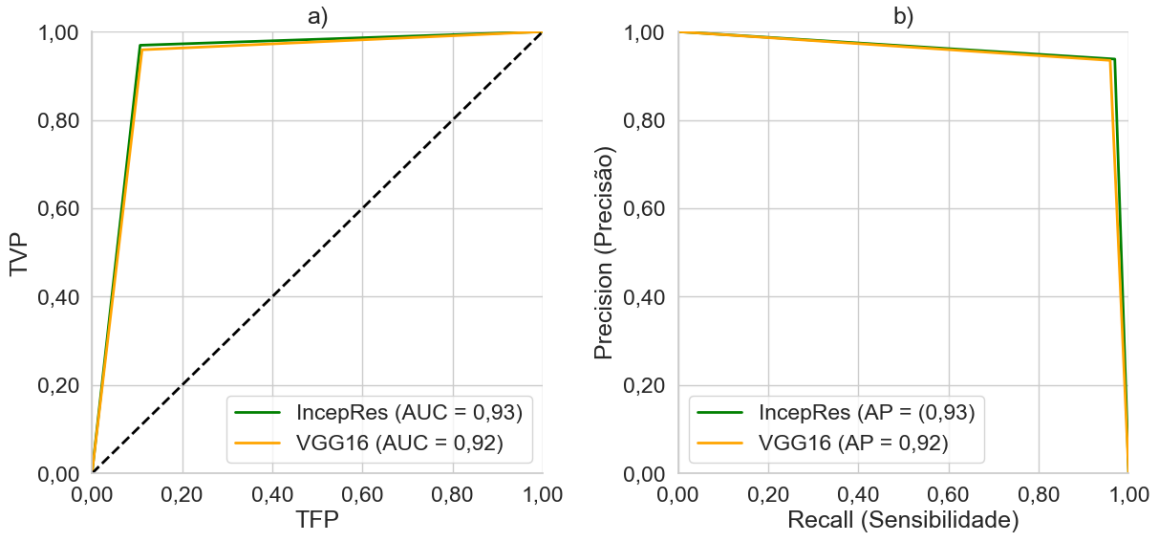
Fonte: Elaborado pelo autor, 2023.

A partir da curva ROC (Gráfico 7a) se nota que dentre os dois modelos, o modelo da Inception-ResNet-V2 apresentou um AUC mais alto, igual a 0,93, em contraste com a VGG-16 que obteve um valor de 0,92. Analisando a curva PR para ambos os modelos (Gráfico 7b) notamos que ambas as redes convolucionais apresentaram um desempenho semelhante entre si,

ao verificarmos a métrica do AP foi constatado um equilíbrio entre os modelos, com uma leve vantagem para o que utiliza a Inception-ResNet-V2.

Diante disso, podemos inferir que ambas as arquiteturas de modelos apresentaram resultados análogos tanto durante o procedimento de treinamento, quanto na etapa final de classificação. Demonstraram ainda uma eficácia equilibrada em sua calibração para distinguir radiografias pediátricas entre aquelas indicativas de pneumonia e aquelas representativas de crianças com pulmões em condição normal.

Gráfico 7 – Curva ROC em relação aos modelos da Inception-ResNet-V2 e VGG-16 (a) e a Curva PR para os dois modelos (b).

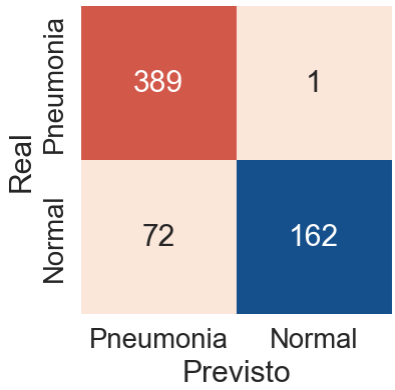


Fonte: Elaborado pelo autor, 2023.

4.2.2 Base de dados balanceada

Ao utilizarmos os modelos treinados com a base de treinamento balanceada, notou-se uma piora significativa na Inception-ResNet-V2 em conseguir classificar de forma exata radiografias normais. A partir da matriz de confusão (Figura 23) para este caso, identifica-se que a precisão do mesmo em classificar corretamente as imagens, como possuindo pneumonia, foi extremante alta, errando apenas uma imagem.

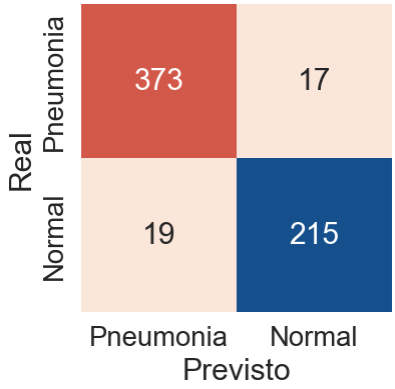
Figura 23 - Matriz de confusão para o modelo utilizando a Inception-ResNet-V2.



Fonte: Elaborado pelo autor, 2023.

Por outro lado, o que foi verificado na avaliação do modelo da VGG-16 pode ser confirmado com os resultados obtidos nesta etapa de classificação na base de teste. Na Figura 24, a matriz de confusão mostra o equilíbrio que o modelo possui em classificar corretamente as radiografias como Normal, acertando 215 das 234 imagens, já na classificação como Pneumonia, obteve uma assertividade de 373 imagens de um total de 390.

Figura 24 - Matriz de confusão para o modelo utilizando a VGG-16.



Fonte: Elaborado pelo autor, 2023.

Como já observado, o desempenho do modelo da Inception-ResNet-V2 apresentou uma diminuição significativa, refletindo em uma acurácia de 88,3%, inferior àquela alcançada pelo modelo quando treinado com a base de dados desbalanceada. Por outro lado, o modelo VGG-16 registrou um sensível aumento na acurácia, atingindo o valor de 94,23%.

Na Tabela 6, pode-se verificar que o modelo utilizando a Inception-ResNet-V2 apresenta uma sensibilidade máxima com valor igual 1, destacando assim, sua eficácia em

classificar corretamente radiografias que apresentam casos de pneumonia. No entanto, ao examinarmos a especificidade do modelo, percebemos que ele não está devidamente calibrado para classificar satisfatoriamente a categoria Normal, resultando em um desequilíbrio no poder de classificação conforme indicado pela média harmônica dada pelo F1-Score para ambas as categorias.

Por outro lado, na Tabela 7, o modelo da VGG-16 possui resultados mais equilibrados, indicando uma calibração mais eficiente. Observa-se uma equidade na capacidade de classificação tanto para a categoria Pneumonia quanto para a categoria Normal, evidenciada pelos valores de sensibilidade e especificidade, que são, respectivamente, 0,96 e 0,92, além dos valores calculado do F1-score que foram superiores aos do modelo Inception-ResNet-V2.

Tabela 6 - Resultados para o modelo da Inception-ResNet-V2

Classes	Precisão	Sensibilidade	Especificidade	F1-Score
PNEUMONIA	0,84	1,00	-	0,91
NORMAL	0,99	-	0,69	0,82

Fonte: Elaborado pelo autor, 2023.

Tabela 7 - Resultados para o modelo da VGG-16.

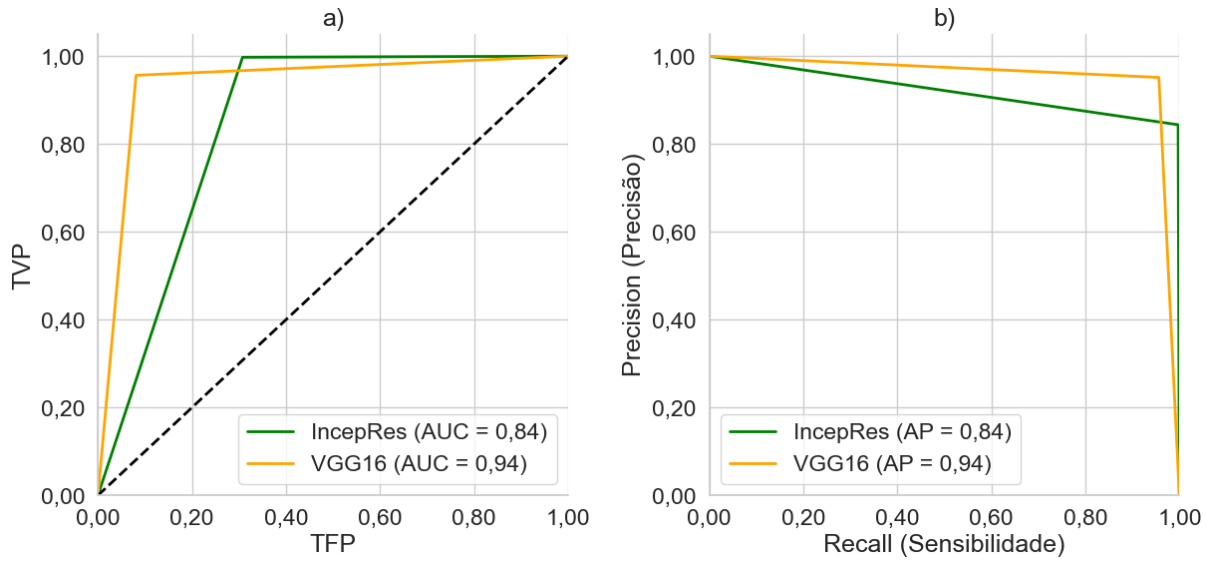
Classes	Precisão	Sensibilidade	Especificidade	F1-Score
PNEUMONIA	0,95	0,96	-	0,95
NORMAL	0,93	-	0,92	0,92

Fonte: Elaborado pelo autor, 2023.

Ao se analisar os valores do AUC de cada modelo, nota-se que o da VGG-16 exibe uma superioridade, evidenciando uma melhoria em comparação com o treinado na base de dados desbalanceada. O AUC atingido foi de 0,94, superando significativamente os 0,84 calculados para o modelo Inception-ResNet-V2 (Gráfico 8a). Este último, quando comparado com o treinado na base desbalanceada, revela uma degradação substancial em seu poder de classificação.

Com base na curva PR (Gráfico 8b) pode-se observar que o modelo da VGG-16 apresenta um equilíbrio e uma superioridade mais evidentes em relação à taxa de precisão e sensibilidade, com um AP igual a 0,94, bem acima dos 0,84 para o outro modelo que embora demonstre uma alta sensibilidade, é comprometido por uma precisão mais baixa.

Gráfico 8 - Curva ROC em relação aos modelos da Inception-ResNet-V2 e VGG-16 (a) e a Curva PR para os dois modelos (b).



Fonte: Elaborado pelo autor, 2023.

4.2.3 Base balanceada × Base desbalanceada

Dessa forma, ao compararmos os modelos treinados com a base de treinamento desbalanceada com aqueles treinados na base balanceada pelo método de *data augmentation*, identificamos dois cenários distintos. No caso da Inception-ResNet-V2, observamos um desempenho superior na base desbalanceada, alcançando um maior equilíbrio ao analisarmos sua sensibilidade e especificidade, porém, quando treinado na base balanceada apresentou um desempenho inferior (Tabela 8).

Por outro lado, o cenário para o modelo da VGG-16 foi mais positivo, evidenciando melhorias a partir do treinamento com a base balanceada. Notamos valores mais elevados de AUC e AP em comparação com o modelo treinado na base sem balanceamento. Além disso, a especificidade aumentou de 89% para 92%, bem como a acurácia que saiu de 94,07% para 94,23% (Tabela 8).

Considerando estes resultados, ao indicar o modelo com o melhor desempenho na fase de classificação e no processo de aprendizagem durante o treinamento, opta-se por aquele que apresenta resultados mais equilibrados, minimizando a propensão a erros, como classificar erroneamente uma radiografia normal como sendo de pneumonia, ou vice-versa, caracterizando os erros do tipo I e do tipo II, respectivamente.

Portanto, o modelo que demonstrou esse equilíbrio e eficácia foi o VGG-16 treinado na base balanceada, com a maioria de suas métricas de avaliação superando as dos demais modelos, exceto pela sensibilidade, que ficou abaixo do modelo Inception-ResNet-V2 treinada em ambas as bases de imagens.

Tabela 8 – Comparação entre os modelos Inception-ResNet-V2 e VGG-16 treinados com a base desbalanceada e balanceada com a aplicação do *data augmentation*.

Modelos	AUC	AP	Acurácia	Sensibilidade	Especificidade
Base Desbalanceada					
Inception-ResNet-V2	93%	93%	93,26%	97%	89%
VGG-16	92%	92%	94,07%	96%	89%
Base Balanceada					
Inception-ResNet-V2	84%	84%	88,3%	100%	69%
VGG-16	94%	94%	94,23%	96%	92%

Fonte: Elaborado pelo autor, 2023.

4.2.4 Comparação com modelos da literatura

Existem vários trabalhos que utilizaram esta mesma base de dados para aplicação e classificação por meio de CNN utilizando o método do *transfer learning*. Para comparação dos resultados apresentados na literatura, destaca-se o modelo de detecção criado por Kermany, Goldbaum, *et al*. (2018) em que utilizaram a arquitetura Inception V3, aplicando o retreinamento apenas na camada final, já o segundo modelo é o proposto por Sousa (2018) que se utilizou da biblioteca *Hyperopt* para realizar uma busca da melhor arquitetura para a resolução do problema em questão.

Em ambos os casos, os modelos passaram por um treinamento em um total de 100 épocas e utilizaram a base de treinamento desbalanceada, contudo, o melhor modelo proposto neste trabalho, obteve resultados semelhantes com apenas 10 épocas de treinamento e utilizando a base de dados balanceada pelo método do *data augmentation*. Para este fim, denominou-se o modelo da VGG-16, que foi identificado com a melhor eficiência em comparação com os demais utilizados neste trabalho, de modelo do autor.

Na Tabela 9, nota-se que em relação ao AUC calculado o modelo proposto por Kermany, Goldbaum, *et al*. (2018) apresentou um valor de 96,8%, acima dos 94% do modelo proposto por Sousa (2018) e o modelo do autor. Em relação a acurácia o modelo de Sousa

(2018) apresentou resultados relativamente melhores, um valor de 95,3%, resultando em uma diferença de 1,07% ao modelo do autor que conseguiu uma acurácia de 94,23%.

O método do autor consegue, mesmo treinado em apenas um décimo de épocas em relação aos demais modelos, obter resultados bastantes satisfatórios que podemos dizer que possui resultados bem semelhantes em relação aos demais. No caso da especificidade, este modelo conseguiu o melhor valor dentre todos sendo igual a 92%, o que demonstra um equilíbrio maior deste modelo em conseguir classificar as radiografias infantis como tendo ou não pneumonia.

Tabela 9 – Comparação de resultados entre o modelo do autor em relação a outros dois modelos presentes na literatura.

Modelos	AUC	Acurácia	Sensibilidade	Especificidade
Modelo do autor	94%	94,23%	96%	92%
Sousa (2018)	94%	95,3%	99,7%	88%
Kermany, *et al*. (2018)	96,8%	92,8%	93,2%	90,1%

Fonte: Elaborado pelo autor, 2023.

5 CONCLUSÃO

A partir das análises realizadas, constatou-se que os dois modelos empregados demonstraram desempenhos bastante similares quando treinados utilizando a base original de imagens de radiografias pediátricas. A Inception-ResNet-V2 obteve algumas métricas levemente superiores as da VGG-16, como um AUC de 0,93. Entretanto, após o balanceamento das imagens de treinamento, a VGG-16 apresentou uma melhoria significativa, enquanto a Inception-ResNet-V2 teve uma queda de desempenho.

Portanto, o modelo VGG-16 foi o modelo que se mostrou mais equilibrado tanto com a base original quanto na base que sofreu o processo de *data augmentation.* Por se tratar de uma arquitetura mais simples que a da Inception-ResNet-V2, se adequou mais aos dados considerando um tempo de treinamento de apenas 10 épocas. Por outro lado, a Inception-ResNet-V2 enfrentou desafios durante o treinamento, especialmente com a base balanceada, apresentando indícios de *overfitting*.

Quanto à eficácia da utilização do *data augmentation*, quanto as transformações geométricas utilizadas, os modelos apresentaram resultados distintos. Enquanto um deles experimentou uma melhoria considerável nas métricas de avaliação, evidenciando a sua capacidade de generalização e robustez diante de variações nos dados de entrada, o outro apresentou uma redução em seu poder de generalização na classificação das classes estudadas, sugerindo uma possível sensibilidade a determinadas transformações. Ressaltando, assim, a importância de escolha de técnicas de transformações geométricas para o balanceamento dos dados de acordo com diferentes tipos de arquiteturas.

O modelo que obteve o melhor desempenho no processo de classificação neste estudo, com a base de imagens de radiografias pediátricas, em comparação com os modelo de Sousa (2018) e de Kermany, Goldbaum, *et al*. (2018), se mostrou mais equilibrado em relação a especificidade e sensibilidade. Tal equilíbrio, em um modelo treinado em apenas 10 épocas em comparação com outros dois que foram treinados em 100 épocas, ressalta a da VGG-16 quando treinada com uma base balanceada pelo método do *data augmentation*.

Neste caso, a aplicação da técnica do *transfer learning* se mostrou eficaz, ao invés do tempo e recursos que seriam gastos para a construção e treinamento de uma CNN do zero, diminuiu-se significativamente a complexidade deste processo e o tempo gasto para o treinamento e aplicação das redes neurais com objetivo de classificar as imagens. Com base nisso, a utilização desse método para auxiliar nos diagnósticos diários de doenças, por meio da classificação de radiografias do tórax de pacientes, demonstrou ser eficaz e eficiente.

REFERÊNCIAS

ABADI, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. **Software available from tensorflow.org**. Disponivel em: <https://www.tensorflow.org/>.

ALMEIDA, R. D.; MAGRINI, A. A matemática das imagens digitais como recurso didático na escola básica. **C.Q.D. - Revista Eletrônica Paulista de Matemática**, Bauru, v. 21, dez. 2021. ISSN 2316-9664.

ALOYSIUS, N.; GEETHA, M. **A review on deep convolutional neural networks**. 2017 international conference on communication and signal processing (ICCSP). Chennai, India: IEEE. 2017. p. 0588-0592.

ANDRADE, F. D. S. **Imagens digitais e matrizes: uma aplicação no ensino médio**. Tese (Mestrado Profissional em Matemática em Rede Nacional) - Universidade Estadual do Sudoeste da Bahia. Vitória da Conquista. 2020.

BEAR, M. F.; CONNORS, B. W.; PARADISO, M. A. Neurônios e Glia. In: **Neurociências:** desvendando o sistema nervoso. 4ª. ed. Porto Alegre: Artmed editora, 2017. p. 23-54. ISBN 978-85-8271-433-1.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BRAGA, A. D. P.; CARVALHO, A. P. D. L. F. D.; LUDERMIR, T. B. **Redes Neurais Artificiais:** Teoria e Aplicações. Rio de Janeiro: LTC Editora, 2000.

BUSLAEV, et al. Albumentations: Fast and Flexible Image Augmentations. **Information**, v. 11, n. 2, p. 125, 2020.

CARDON, A.; MÜLLER, D. N.; NAVAUX, P. Introdução às redes neurais artificiais, Porto Alegre, 1994.

CHOLLET, F. Keras, 2015. Disponivel em: <https://github.com/fchollet/keras>.

EBERHART, R. C.; DOBBINS, R. W. **Neural Network PC Tools:** A Practical Guide. San Diego: Academic Press, 1990.

FAWCETT, T. An introduction to ROC analysis, v. 27, n. 8, p. 861-874, 2006.

FLORES, D. Escola educação. **Sinapses**, 17 Abril 2020. Disponivel em: <https://escolaeducacao.com.br/sinapses/>. Acesso em: 04 set. 2023.

GÉRON, A. Treinando Modelos. In: GÉRON, A. **Mãos à Obra:** Aprendizado de Máquina com Scikit-Learn & TensorFlow. Rio de Janeiro: Alta Books, 2019. Cap. 4, p. 109-148.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, v. 585, n. 7825, p. 357-362, Setembro 2020. ISSN 10.1038/s41586-020-2649-2. Disponivel em: <https://doi.org/10.1038/s41586-020-2649-2>.

HAYKIN, S. **Redes neurais:** princípios e prática. 2ª. ed. Porto Alegre: Bookman, 2001.

HE, K. et al. Deep residual learning for image recognition. **Proceedings of the IEEE conference on computer vision and pattern recognition**, p. 770-778, 2016.

HUNTER, J. D. Matplotlib: A 2D graphics environment. **Computing in Science & Engineering**, v. 9, n. 3, p. 90-95, 2007.

IDE, H.; KURITA, T. **Improvement of learning for CNN with ReLU activation by sparse regularization**. 2017 International Joint Conference on Neural Networks (IJCNN). Anchorage, ASK, USA: IEEE. 2017. p. 2684-2691.

IZADYYAZDANABADI, M. et al. Convolutional neural networks: Ensemble modeling, fine-tuning and unsupervised semantic localization for neurosurgical CLE images. **Journal of Visual Communication and Image Representation**, v. 54, p. 10-20, 2018. ISSN 1047-3203.

KERMANY, D. S. et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. **cell**, v. 172, n. 5, p. 1122-1131, 2018.

KORKMAZ, F. T.; TRABER, K. E. Innate immune responses in pneumonia. **Pneumonia**, v. 15, n. 1, p. 1-26, 2023. ISSN 2200-6133. Disponivel em: <https://doi.org/10.1186/s41479-023-00106-8>.

LECUN, Y. et al. Gradient-based Learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278-2324, 1998.

LIN, M.; CHEN, Q.; YAN, S. Network in network. **arXiv preprint arXiv:1312.4400**, 2013.

MARTINS, A. P. **Aplicação de redes convolucionais profundas para a detecção de massas em mamografias**. TCC (Graduação) - Engenharia Eletrônica, Centro Tecnológico, Universidade Federal de Santa Catarina. Florianópolis, p. 100. 2019.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, p. 115-133, 1943.

MCKINNEY, W. Data Structures for Statistical Computing in Python. **Proceedings of the 9th Python in Science Conference**, v. 445, p. 56-61, 2010.

NIED, A. **Treinamento de redes neurais artificiais baseado em sistemas de estrutura variável com taxa de aprendizado adaptativa**. Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Minas Gerais. Belo Horizonte. 2007.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825-2830, 2011.

RAHMAT , T.; ISMAIL, A.; ALIMAN, S. Chest X-Rays Image Classification in Medical Image Analysis. **Applied Medical Informatics**, v. 40, n. 3-4, p. 63-73, 2018. Disponivel em: <https://ami.info.umfcluj.ro/index.php/AMI/article/view/639>.

RAMCHOUN, H. E. A. Multilayer perceptron: Architecture optimization and training, 2016.

SHORTEN, C.; KHOSHGOFTAAR, M. A survey on image data augmentation for deep learning. **Journal of big data**, v. 6, n. 1, p. 1-48, 2019.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SOUSA, G. G. B. **Deep learning para a detecção e classificação de pneumonia por radiografias do tórax**. TCC (Bacharelado em Ciência da Computação) - Universidade Federal do Maranhão. São Luís. 2018.

SZEGEDY, et al. Inception-v4, inception-resnet and the impact of residual connections on learning. **Proceedings of the AAAI conference on artificial intelligence**, v. 31, n. 1, 2017.

TAJBAKHSH, N. et al. Convolutional neural networks for medical image analysis: Full training or fine tuning? **IEEE transactions on medical imaging**, v. 35, n. 5, p. 1299-1312, Maio 2016.

TISSOT, H. C.; CAMARGO, L. C.; POZO, A. T. R. Treinamento de redes neurais feedforward: comparativo dos algoritmos backpropagation e differential evolution. **Brazilian Conference on Intelligent Systems**, 2012.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. **Proceedings of the xxix conference on graphics, patterns and images**, v. 1, n. 4, 2016.

VGG16 - Convolutional Network for Classification and Detection. **Neurohive.io**, 2018. Disponivel em: <https://neurohive.io/en/popular-networks/vgg16/>. Acesso em: 20 out. 2023.

VOGADO, L. H. S. et al. **Rede neural convolucional para o diagnóstico de leucemia**. Anais do XIX Simpósio Brasileiro de Computação Aplicada à Saúde. Porto Alegre: Sociedade Brasileira de Computação. 2019. p. 46-57. ISSN 2763-8952.

WANG, M. et al. **A high-speed and low-complexity architecture for softmax function in deep learning**. 2018 IEEE asia pacific conference on circuits and systems (APCCAS). : IEEE. 2018. p. 223-226.

WASKOM, M. L. seaborn: statistical data visualization. **The Open Journal**, v. 6, n. 60, p. 3021, 2021. Disponivel em: <https://doi.org/10.21105/joss.03021>.

WORLD HEALTH ORGANIZATION. Pneumonia. **www.who.int**, 2019. Disponivel em: <https://www.who.int/health-topics/pneumonia#tab=tab_1>. Acesso em: 10 nov. 2023.

WORLD HEALTH ORGANIZATION. Pneumonia in children. **www.who.int**, 2022. Disponivel em: <https://www.who.int/news-room/fact-sheets/detail/pneumonia#:~:text=Pneumonia%20is%20the%20single%20largest,aged%201%20to%205%20years.>. Acesso em: 10 nov. 2023.

YOSINSKI, J. et al. How transferable are features in deep neural networks?. **Advances in neural information processing systems**, v. 27, 2014.

ZHOU, et al. Learning deep features for discriminative localization. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 2921-2929, 2016.