UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE FÍSICA

CLASSIFICAÇÃO DE PNEUMONIA EM RADIOGRAFIA DO TÓRAX UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

HENRIQUE HUNDSDORFER VEDOVELI

ORIENTADOR: Prof. Dr. Anuar José Mincache
COORIENTADOR: Prof. Dr. Breno Ferraz de Oliveira

MARINGÁ
2022

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE FÍSICA

HENRIQUE HUNDSDORFER VEDOVELI

CLASSIFICAÇÃO DE PNEUMONIA EM RADIOGRAFIA DO TÓRAX UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Departamento de Física da Universidade Estadual de Maringá, sob orientação dos professores, Prof. Dr. Anuar José Mincache e Prof. Dr. Breno Ferraz de Oliveira, como parte dos requisitos necessários para obtenção do título de Bacharel em Física.

Orientador: Prof. Dr. Anuar José Mincache

Coorientador: Prof. Dr. Breno Ferraz de Oliveira

Maringá, Abril de 2022

HENRIQUE HUNDSDORFER VEDOVELI

CLASSIFICAÇÃO DE PNEUMONIA EM RADIOGRAFIA DO TÓRAX UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Orientador: Prof. Dr. Anuar José Mincache
Universidade Estadual de Maringá - UEM

Profª. Dra. Lilian Felipe da Silva Tupan
Universidade Estadual de Maringá - UEM

Prof. Dr. Paulo Henrique Soares
Universidade Tecnológica Federal do Paraná
- UTFPR

Maringá, Abril de 2022

Agradecimentos

Meus sinceiros agradecimentos ao Prof. Dr. Anuar José Mincache e ao Prof. Dr. Breno Ferraz de Oliveira pela orientação e pelos ensinamentos ao decorrer do curso, e à todos professores que de alguma maneira contribuíram com minha caminhada.

Agradeço à todos novos e velhos amigos, pelo apoio, ensinamentos e momentos de diversão em especial ao Felipe Bono, Guilherme Minister, Caio Costa, Alexandre, Ana, Hugo, Hygor, Luiz, Nathan, Thais e Victoria.

E por último, mas não menos importante, gostaria de agradecer aos meus pais, Eloisa e Wanderley, por tudo que já fizeram por mim.

Resumo

A pneumonia afeta grande porcentagem da população mundial e uma das formas de diagnóstico da doença é através da realização da radiografia de tórax, para automatizar o processo de diagnóstico visando diminuir o tempo para obtenção do resultado do exame e reduzir as probabilidades de ocorrência de erros humanos, é proposto um método para classificação automática das imagens provenientes do exame de radiografia do tórax, utilizando redes neurais convolucionais. A escolha do modelo classificatório a ser utilizado foi tomada como base a avaliação do desempenho de arquiteturas de redes neurais já conhecidas, sendo elas a *LeNet-5* (78,80% de acurácia para a classificação multi classe e 86,85% para a classificação binária), *AlexNet* (87,01 % de acurácia para a classificação multi classe e 92,31% para a classificação binária), *VGG-16* (87,02% de acurácia para a classificação multi classe e 95,83% para a classificação binária), *Inception V3* (84,46% de acurácia para a classificação multi classe e 90,87% para a classificação binária) e o modelo *ResNet-50* (84,14% de acurácia para a classificação multi classe e 85,58% para a classificação binária). Utilizando estas redes neurais foi proposto a construção de uma nova rede neural convolucional utilizando características das arquiteturas *VGG-16* e *Inception V3*, denominada *VGG-Inc*, com esta foi-se obtido uma taxa de acerto para a classificação multi-classe de 91,35%, para cada classe foi obtido um F_1-*Score* de 0,947 para a classe de pulmões saudáveis, 0,929 para a classe de pneumonia bacteriana e 0,837 para a classe de pneumonia viral. Para a classificação binária o modelo obteve uma taxa de acerto de 96,15% para os dados reservados ao teste do modelo. Sendo assim, junto ao médico responsável, espera-se que o novo modelo classificatório proposto capaz de classificar a presença da pneumonia e diferenciar as imagens classificadas como pneumonia entre viral e bacteriana diminua consideravelmente o tempo para a decisão médica e diminuindo também as probabilidades de ocorrência do erro humano na hora do diagnóstico.

Palavras-chave: Classificação de Imagens, Radiografia de Tórax, Redes Neurais Convolucionais.

Sumário

1 Introdução

A pneumonia é uma forma de infecção respiratória aguda que atinge os alvéolos pulmonares, causada principalmente por vírus ou bactérias, estima-se que ocorram cerca de 450 milhões de casos por ano em todo o mundo [1] e somente no ano de 2017 foi responsável por 15% das mortes de crianças menores de cinco anos, matando 808.694 crianças [2]. Uma das formas de diagnóstico é a partir de exames de radiografias realizadas na região do tórax, onde o médico responsável pode realizar o diagnóstico e iniciar o tratamento [3], porém o processo de diagnóstico demanda tempo e está suscetível ao erro humano.

Os avanços da tecnologia vem possibilitando formas de automatização que diminuem o tempo até se ter o diagnóstico e a probabilidade da ocorrência do erro humano, os sistemas de diagnóstico auxiliado por computador[1] (*CAD*) foram inicialmente projetados nos anos 1960 e desde então vem sendo utilizado como segunda opinião médica e tornando o processo de resultado de exames mais rápido e preciso [4]. Uma das formas possíveis de se construir um *CAD* é através das redes neurais convolucionais (*CNNs* [2]) [5], sendo um tipo específico de algoritmo de aprendizado profundo projetadas para processar dados em formas de múltiplos *arrays*[3] [6], como é o caso das imagens.

Com o intuito de construir um *CAD* capaz de diminuir o tempo para se obter o resultado do exame e diminuir as probabilidades de ocorrer o erro humano é proposto a construção e treinamento de uma *CNN* capaz de classificar a pneumonia, e caso a imagem seja classificada como pneumonia qual tipo (viral ou bacteriana), nas imagens geradas a partir dos exames de radiografia do tórax. Para realizar o treinamento e o teste do modelo classificatório será utilizado as imagens contidas no conjunto de dados *"Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images"*, que contém imagens rotuladas e validadas de radiografias do tórax de pulmões saudáveis, com pneumonia viral e bacteriana de diferentes pacientes [7].

A escolha da arquitetura da rede neural que será utilizada para o modelo classificatório final será baseada na avaliação do desempenho na classificação de novas imagens de diferentes arquiteturas já conhecidas, sendo elas a *LeNet-5* [8], *AlexNet* [9], *VGG-16* [10], *Inception V3* [11] e *ResNet-50* [12]. Para a realização da avaliação dos modelos treinados serão utilizadas a acurácia, precisão, sensibilidade e o F_1-*score* como métricas de avaliação. Com a escolha da arquitetura de rede neural será realizado o treinamento do modelo final utilizando técnicas de regularização da rede e otimização, com o objetivo de evitar o sobre ajuste do modelo aos dados destinados ao treinamento e aumentar sua capacidade de

[1] Tradução livre para *Computer-Aided Diagnosis*
[2] Abreviação para *Convolutional Neural Networks*
[3] *Array* é a forma computacional de se representar um tensor.

generalização.

Este trabalho está estruturado em cinco capítulos, no Capítulo 2 será realizado a revisão de conceitos teóricos fundamentais com base na literatura para a realização do trabalho, no Capítulo 3 é tratado dos Materiais e Métodos que serão utilizados, já no Capítulo 4 são apresentados os resultados que foram obtidos durante a realização deste trabalho e por fim no Capítulo 5 são apresentadas as conclusões tomadas a partir dos resultados obtidos.

2 Fundamentação Teórica

2.1 Pneumonia e Radiografia de Tórax

A pneumonia é uma doença que afeta os alvéolos pulmonares deixando-os cheio de pus e líquido, tornado assim a respiração dolorosa e limitando a ingestão de oxigênio [1, 2], apesar de a maioria dos efeitos da doença terem sido controlados nos últimos anos devido aos avanços científicos a pneumonia continua sendo uma das principais causa de morte em países em desenvolvimento, as maiores incidências surgem em crianças menores de cinco anos e adultos maiores de 75 anos e somente no ano de 2017 foi responsável pela morte de 808.694 crianças menores que cinco anos [2, 13]. A pneumonia pode ser causada principalmente bactérias, vírus ou por fungos e a capacidade de diferenciar a pneumonia viral da bacteriana pode ter implicações importantes no manejo [13].

Uma das formas de realizar o diagnóstico da doença é através da realização de radiografias de tórax [3], que permite de modo não invasivo adquirir imagens de alta qualidade para servir como registro para a investigação de possíveis alterações da saúde de pacientes sintomáticos ou assintomáticos [14].

Para a realização de uma radiografia de tórax coloca-se uma tela que interage com os raios-X, atrás do paciente, então o paciente recebe um feixe de raios-X na região do pulmão e a partir da absorção dos tecidos a radiação X é possível gerar a imagem [15, 16]. Na Figura 1 temos exemplos de tomografias do tórax.

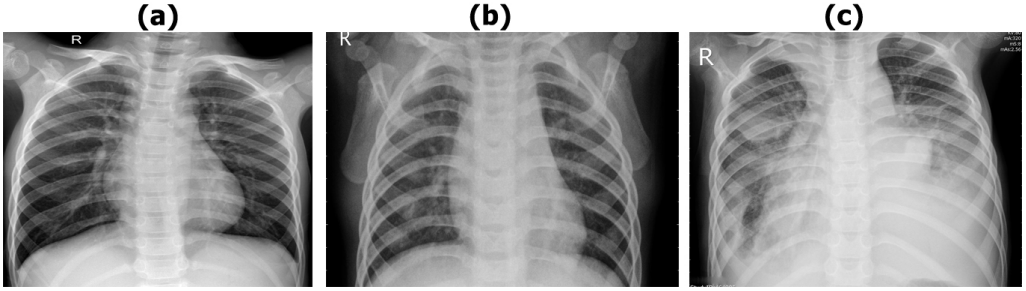

Figura 1 – Radiografia de tórax de **(a)** Pulmão Saudável; **(b)** Pneumonia Viral; **(c)** Pneumonia Bacteriana; Fonte: *Kermany, Zhang* e *Goldbaum* (2018).

Os raios-X interagem diferentemente com os tecidos ou corpos ao colidir, gerando assim tons de cinza diferentes, áreas que contém algum gás, constituem as partes mais escuras das imagens, tecidos adiposos geram uma imagem cinza-escuro, partes compostas por tecidos moles apresentam uma coloração acinzentada e ossos ou cálcio apresentam uma cor branca [16], a partir das imagens geradas no exame o médico responsável pode realizar

o diagnóstico da presença ou não da pneumonia no paciente junto a outras evidências, caso o diagnóstico seja positivo, é possível identificar a natureza da doença.

As áreas mais claras nas imagens 1b e 1c ocorrem devido ao acúmulo de pus nos alvéolos pulmonares e a partir das informações que o médico responsável obtêm ao realizar a análise das imagens do exame de tomografia é possível de se realizar o diagnóstico da doença e iniciar o tratamento.

2.2 Redes Neurais Artificiais

Inicialmente podemos definir inteligência artificial como a área de estudo preocupada em construir programas de computadores inteligentes semelhantes a tarefas humanas, porém não necessariamente precisa limitar-se a métodos que são biologicamente observáveis [17], uma das subáreas da inteligência artificial é o aprendizado de máquina sendo este focado em desenvolver modelos computacionais capazes de melhorar seu desempenho com base em resultados anteriores [18]. Modelos computacionais de aprendizado de máquina são amplamente utilizados para realizar previsões com base em dados existentes, além de processamento de linguagem natural[1], processamento de imagens, entre outras aplicações possíveis [19].

Os modelos de aprendizado profundo, também conhecidos como redes neurais artificiais (RNAs) são sistemas computacionais inspirados na forma em que o sistema biológico opera, sendo uma forma mais complexa de algoritmo de aprendizado de máquina [20], e possibilitam modelos computacionais compostos por muitas camadas de processamento de dados e parâmetros ajustáveis aprender as representações de dados com vários níveis de abstração, conseguindo assim exceder o desempenho de formas anteriores de inteligência artificial, como o aprendizado de máquina [6, 19, 20] a popularidade de modelos de aprendizado profundo vem aumentando nos últimos anos, já que o aumento da capacidade de processamento dos computadores vem crescendo, possibilitando assim resolver problemas mais complexos que modelos de aprendizado de máquina tradicional.

A forma elementar de uma RNA é o neurônio matemático[2] (Figura 2), que recebe a soma de x_n entradas multiplicadas a w_n pesos sinápticos, então são somados a um viés b, a soma das entradas pesadas com o viés é utilizada como argumento para uma função não linear $f(\cdot)$ que retorna o nível de excitação de cada neurônio [21].

$$\hat{y}(x, w) = f\left(\sum_{i=1}^{n} x_i w_i + b\right) \tag{2.1}$$

[1] Tradução livre para Natural Language Processing

[2] Daqui em diante o neurônio matemático será referido apenas como neurônio.

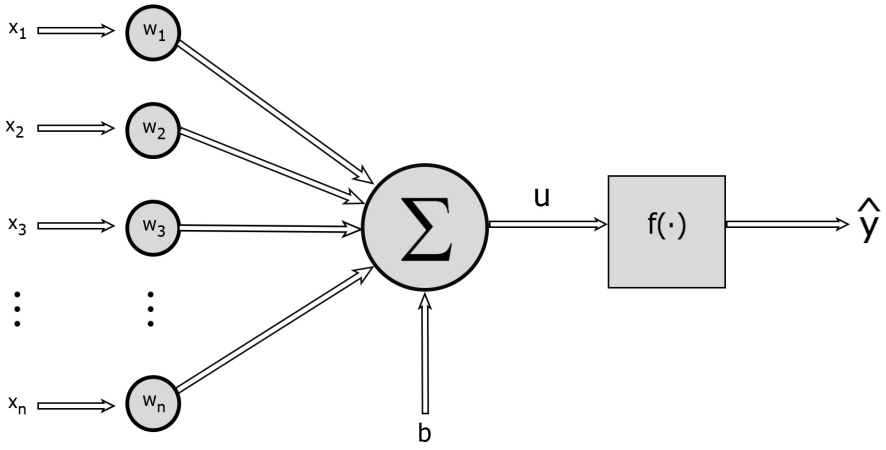

Figura 2 – Representação gráfica de um neurônio matemático. Adaptado de *Aggarwal* (2018).

Podemos representar a 2.1 de forma matricial:

$$\hat{y}(\mathbf{X}, \mathbf{W}) = f(\mathbf{W}^T\mathbf{X} + b) \tag{2.2}$$

onde $\mathbf{W}$ é o vetor contendo os pesos sináptico, $\mathbf{X}$ os valores de entrada, a partir de ajustes dos valores dos pesos sinápticos é possível encontrar o melhor resultados. As Equações 2.1 e 2.2 são a representação de uma camada computacional única[3], ou perceptron [22, 23].

É possível aumentar a capacidade de generalização de uma RNA aumentando o número de camadas computacionais que contidas no modelo, quando uma rede neural contém duas ou mais camadas ela é denominada perceptron de múltiplas camadas[4] (MLP) (Figura 3), a primeira camada é chamada de camada de entrada e a última é chamada de camada de saída, as camadas entre elas são as camadas ocultas [23].

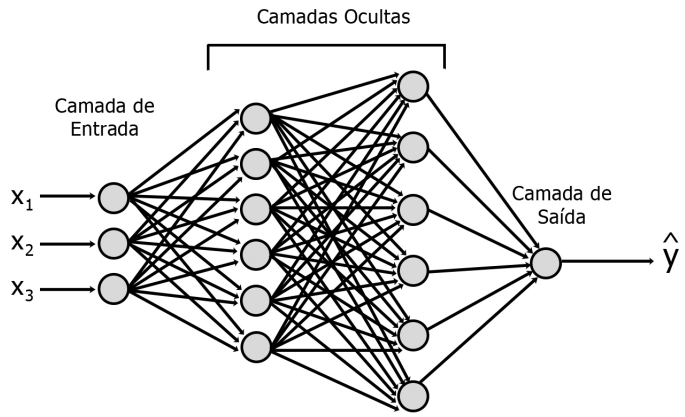

Figura 3 – Estrutura de uma MLP com duas camadas ocultas, cada círculo representa um neurônio (Figura 2). Adaptado de *Aggarwal* (2018).

[3] Tradução livre para *Single Computational Layer.*
[4] Tradução livre para *Multilayer Perceptron.*

2.2.1 Funções de Ativação

As funções de ativação como são chamadas as funções não lineares $f(\cdot)$, que ao receber a soma dos valores de entrada associados com seus pesos e o viés podem ou não excitar um neurônio, e são geralmente utilizadas para aumentar a capacidade de generalização da rede, sendo o núcleo de uma estrutura de uma RNA [24]. A seguir serão enunciadas as funções de ativação ReLU (*Rectified Linear Unit*), *TanH* (Tangente Hiperbólica) e *softmax*.

2.2.1.1 *ReLU*

A função *Rectified Linear Unit* ou ReLU pode ser descrita matematicamente pela Equação 2.3, sendo amplamente utilizadas como função de ativação para as camadas ocultas de uma RNA, pois são funções de fácil otimização [25], porém para quando recebe como parâmetro um padrão não excitatório a função retorna zero como resposta [26], tornando assim o processo de treinamento menos custoso computacionalmente.

$$f(u) = max(0, u) \tag{2.3}$$

sendo u é a soma das entradas pesadas com o viés. A Equação 2.3 pode ser reescrita da seguinte maneira:

$$f(u) = \begin{cases} u \ ; \ se \ u > 0 \\ 0 \ ; \ se \ u \ \leq 0 \end{cases} \tag{2.4}$$

Sendo assim o gráfico para a função em um domínio de $[-10, 10]$ pode ser representado pela Figura 4.

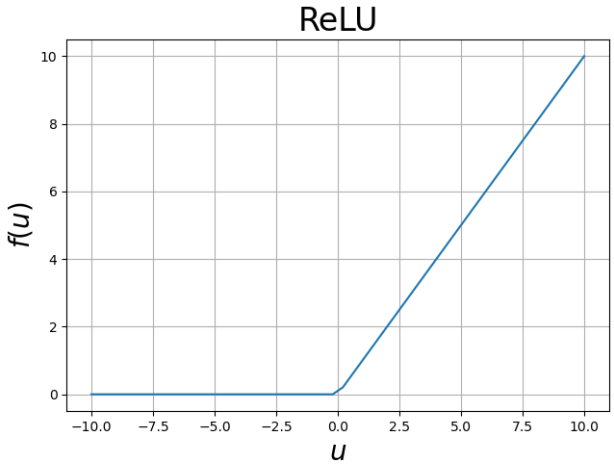

Figura 4 – Gráfico para a função ReLU. Fonte: O Autor

2.2.1.2 *TanH*

A função de ativação tangente hiperbólica ou apenas *tanH* é uma função que tem a forma de um "S" estando contida em y no intervalo $[-1, 1]$ e tendo 0 como a raiz da função, por ser de otimização relativamente fácil, assim como a função *ReLU* a função *tanH* pode ser utilizada como função de ativação para as camadas ocultas de uma RNA [25]. Podemos definir a função tangente hiperbólica matematicamente como:

$$f(u) = tanH(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \tag{2.5}$$

sendo u é a soma das entradas pesadas com o viés.

Utilizando a Equação 2.5 podemos construir o gráfico para a tangente hiperbólica em um domínio de [-10,10], dado pela Figura 5:

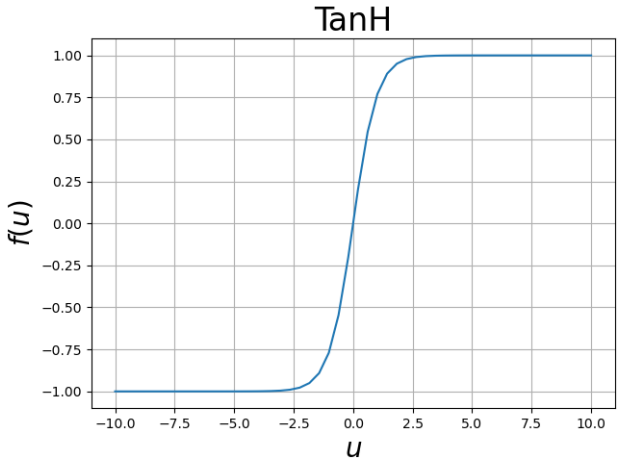

Figura 5 – Gráfico para a função *tanH*. Fonte: O Autor

2.2.1.3 *Softmax*

A função *softmax* assim como a tangente hiperbólica tem um formato de "S", porém esta é utilizada na camada de saída de uma RNA para problemas de classificação com n classes, recebendo como parâmetro a soma dos valores de entrada associados com seus pesos sinápicos e com o viés e retorna a probabilidade sobre uma variável discreta com n possíveis valores [25, 27], representada pela Equação 2.6:

$$P(y|u_i) = soft(u)_i = \frac{e^{u_i}}{\sum_{k=1}^{n} e^{u_k}} \tag{2.6}$$

A partir da Equação 2.6 é possível determinar a probabilidade de u ser pertencente a classe y, utilizando a função é possível também gerar o gráfico que representa a distribuição de probabilidade de u pertencer a y representado na Figura 6.

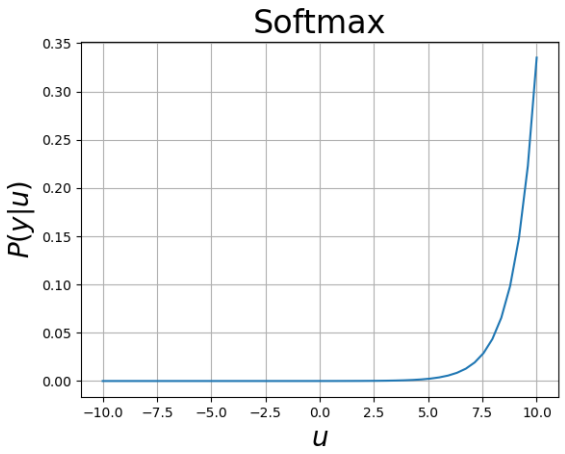

Figura 6 – Gráfico para a função *softmax*. Fonte: O Autor

Quando calculamos a função *softmax* para duas classes, o que seria o caso de uma classificação binária, podemos considerar que os valores passados como parâmetro para a função *softmax* sendo os valores arbitrários $[a, b]$ utilizando a Equação 2.6 podemos encotrar a probabilidade destes valores pertencerem à classe y.

$$soft(u)_1 = \frac{e^{u_1}}{\sum_{k=1}^{2} e^{u_k}}$$

somando os valores da somatória,

$$soft(u)_1 = \frac{e^{u_1}}{e^{u_1} + e^{u_2}}$$

como $u_1 = a$ e $u_2 = b$, temos que

$$soft(u)_1 = \frac{e^{a}}{e^{a} + e^{b}}$$

dividindo o numerador e o denominador por e^a:

$$soft(u)_1 = \frac{\frac{e^a}{e^a}}{\frac{e^a}{e^a} + \frac{e^b}{e^a}}$$

$$soft(u)_1 = \frac{1}{1 + e^{b-a}}$$

sendo essa uma equação subdeterminada, podemos simplifica-la assumindo que $b = 0$, portanto temos que:

$$P(y|a) = \frac{1}{1 + e^{-a}} \tag{2.7}$$

analogamente a probabilidade de b pertencer a classe y é dada pela equação:

$$P(y|b) = \frac{1}{1 + e^{-b}} \tag{2.8}$$

As Equações 2.7 e 2.7 são casos particulares da função *softmax* quando temos apenas duas classes, para esse caso a equação *softmax* recebe o nome de *sigmoide* ou função logística.

2.3 Redes Neurais Convolucionais

Dentro das formas possíveis de algoritmos de aprendizado profundo temos a redes neurais convolucionais (*CNNs*) que são análogas as *RNAs* tradicionais, pois são compostas por parâmetros que se auto-optimizam por meio da experiência [20], porém para uma rede neural artificial ser considerada uma rede neural convolucional deve conter em sua arquitetura as camadas de convolução, *pooling* e *dense*. Essa característica faz com que as redes neurais convolucionais sejam amplamente utilizadas para processar dados em formas de múltiplos *arrays* [6], como é o caso das imagens.

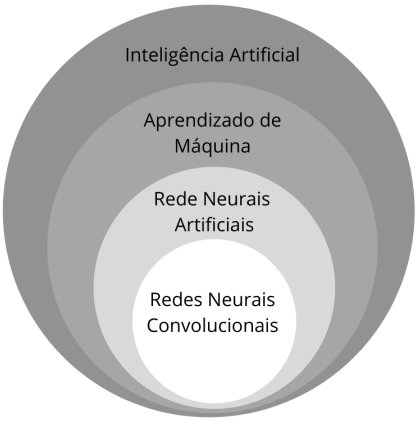

Figura 7 – Diagrama de *Venn* relacionando as subáreas da inteligência artificial. Adaptado de *Goodfellow*, *Bengio* e *Courville* (2016).

Em 1998 foi proposto por *Yann Lecun et al.* o modelo de rede neural convolucional *LeNet-5* para a identificação de dígitos manuscritos de código postal no serviço postal, o modelo proposto tinha um total de 60.000 parâmetros treináveis [8]. Por alguns anos não ocorreram avanços significativos neste campo de estudo, porém em 2012 *Alex Krizhevsky et al.* desenvolveram um modelo similar a *LeNet-5* para a competição *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC), o modelo denominado *AlexNet* continha cerca de 60 milhões de parâmetros treináveis [9] e obteve a melhor taxa de acerto da competição ILSVRC-2012, convencendo assim a comunidade cientifica da capacidade das *CNNs*.

Utilizando os conceitos propostos por *Yann Lecun et al.* e *Alex Krizhevsky et al.* em 2014 foi proposto por Karen *Simonyan e Andrew Zisserman* para a ILSVRC-2014 um modelo contendo um total de 138 milhões de parâmetros treináveis e 16 camadas contendo parâmetros treinaveis *VGG-16* [10]. Na mesma competição foi proposto outro modelo, com maior profundidade mais mantendo o custo computacional, chamado *Inception* [11]. Um dos últimos avanços que ocorreram no campo de redes neurais convolucionais foi a proposta do modelo *ResNet* por *Kaiming Hen et al.* no final de 2015, o modelo obteve a melhor taxa de acerto na competição ILSVRC-2015.

Cada modelo será melhor estudado na Seção 2.3.2.

2.3.1 Estrutura Básica de uma CNN

A estrutura básica de uma *CNN* é composta pela camada de entrada, camada de convolução, camada de *pooling* e pela camada *dense* ou totalmente conectada. Nas próximas seções serão apresentadas as camadas que compões a estrutura básica de uma rede neural convolucional.

2.3.1.1 Camada de Entrada

Apesar de um computador não conseguir interpretar uma imagem como os seres humanos, podemos representar uma imagem em escala de cinza, por exemplo, como o caso da Figura 8, como uma função $I(x, y)$, de modo que seja possível o computador reconhece-la, onde os argumentos x e y são coordenadas espaciais e o valor da função I é a intensidade de cinza no ponto (x, y) da imagem [28].

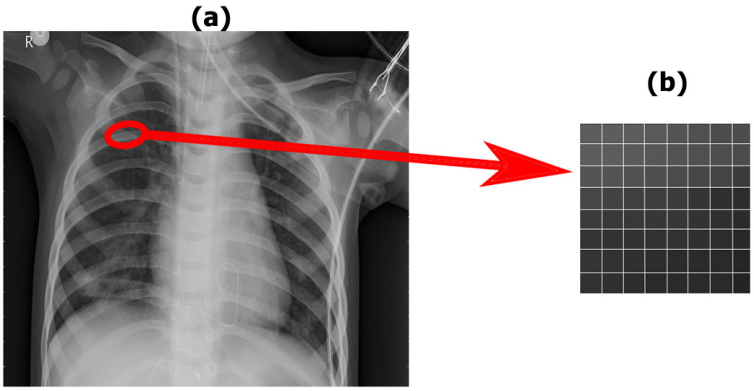

Figura 8 – **(a)** Radiografia do tórax, **(b)** Ampliação da imagem (a); Adaptado de *Kermany, Zhang* e *Goldbaum* (2018).

Sendo assim, podemos representar a imagem 8b como uma matriz, onde o elemento a_{ij} representará o valor de $I(i, j)$, o valor para a intensidade de cinza em cada pixel da Figura 8b está representado na matriz da Equação 2.9.

$$I(i,j) = \begin{bmatrix} 92 & 94 & 94 & 90 & 83 & 81 & 79 & 77 \\ 92 & 91 & 91 & 88 & 84 & 82 & 80 & 76 \\ 87 & 83 & 81 & 80 & 76 & 82 & 68 & 64 \\ 72 & 67 & 66 & 65 & 61 & 56 & 52 & 48 \\ 63 & 60 & 60 & 61 & 58 & 54 & 51 & 48 \\ 55 & 53 & 54 & 54 & 52 & 50 & 48 & 46 \\ 54 & 52 & 51 & 50 & 48 & 48 & 47 & 43 \\ 54 & 51 & 48 & 46 & 45 & 47 & 45 & 41 \end{bmatrix} \tag{2.9}$$

Portanto, a camada inicial de uma arquitetura de uma *CNN* para processamento de imagem é análoga à camada de entrada de uma RNA, de modo que a matriz contendo os valores de intensidade de cada pixel da imagem representa um valor nos valores de entrada para a rede neural convolucional [20, 29], ou seja cada elemento da matriz 2.9 será uma componente do vetor $\mathbf{X}$ na Equação 2.1.

Uma imagem em escala de cinza representa o caso específico para quando a imagem apresenta apenas um canal de cor, quando temos uma imagem colorida, por exemplo, podemos representa-la no sistema de cores RGB, tendo assim três matrizes ao invés de apenas uma onde cada matriz representará a intensidade de vermelho, verde e azul em cada ponto.

2.3.1.2 Camada de Convolução

Uma das camadas obrigatórias nas arquiteturas de redes neurais convolucionais são as camadas de convolução, seu nome é devido à operação de convolução realizada nesta camada, a convolução é dada pela integral que expressa a quantidade de sobreposição de uma função g quando deslocado em relação a outra função f [30], denominamos f como a matriz de entrada e g como a matriz de convolução ou *kernel*. Portanto, temos que a convolução entre f e g ao longo do intervalo $[-\infty, +\infty]$ é definida matematicamente por:

$$[f * g] = \int_{-\infty}^{\infty} f(\tau)g(x - \tau)d\tau \tag{2.10}$$

Quando trabalhamos com imagens não temos um intervalo infinito, mas sim um intervalo finito e bidimensional, então podemos reescrever a Equação 2.10 como o produto escalar entre as duas matrizes e a soma dos valores gerados, cujo resultado será armazenado em uma nova matriz [31]. De modo que a Equação 2.11 representa a convolução discreta em duas dimensões:

$$[f(x, y) * g(x, y)] = \sum_{m} \sum_{n} f(m, n)g(x - m, y - n) \tag{2.11}$$

Para implementar computacionalmente uma camada de convolução em uma *CNN* é necessário utilizar alguns valores de parâmetros para o funcionamento da camada, sendo eles o tamanho da matriz de convolução que será convolucionada com a imagem, os valores de cada elemento do *kernel* serão ajustados durante o processo de treinamento. Outro valor de parâmetros necessário é o número de filtros de dimensionalidade, que serão o número de matrizes de convolução que serão utilizados em cada convolução, podendo assim alterar a dimensionalidade da matriz resultante, por exemplo, uma matriz de tamanho (226x226x1) que sofre uma convolução com 10 matrizes de convoluções terá como matriz resultante uma matriz de tamanho (226x226x10) [32], é necessário também passar como

parâmetro o número de pixels que o *kernel* se deslocará na imagem, chamado *stride*, por fim é necessário definir a função de ativação que será aplicada na matriz resultante.

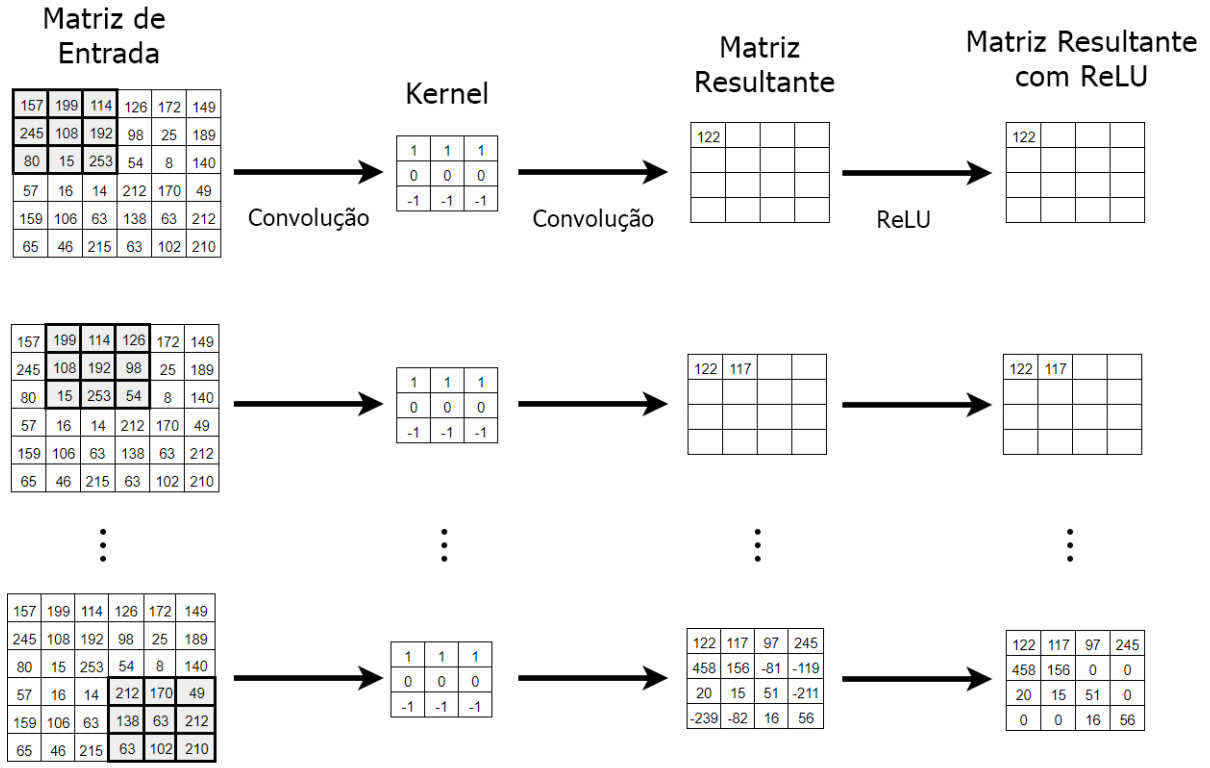

Figura 9 – Convolução com *stride* de (1x1) em uma matriz de tamanho (6x6) utilizando um *kernel* de tamanho (3x3) com valores arbitrários, na matriz resultante à convolução foi aplicada a função de ativação ReLU; Fonte: O Autor.

2.3.1.3 Camada de *Pooling*

Durante a camada de *pooling* ocorre uma redução da dimensionalidade do *array* passado como entrada, reduzindo assim a complexidade computacional do modelo [20], a função de *pooling* utilizada substitui os valores de um certo local da imagem, utilizando uma função para resumir o intervalo do *array* [25]. Para definir uma camada de *pooling* computacionalmente é necessário passar os valores de alguns parâmetros, como o tamanho do intervalo a qual a função de *pooling* vai ser aplicada, este intervalo também é conhecido como janela de *pooling*, além deste valor é necessário passar o valor de quantos pixels a janela de *pooling* se movimentará pelo *array*, este parâmetro é chamado *stride*.

Existem diferentes tipos de funções de *pooling* que podem ser utilizadas para construir uma rede neural convolucional, porém as mais utilizadas são a *Max Pooling* e *Average Pooling*. A função *Max Pooling* nos retorna dentro da janela de *pooling* apenas o maior valor encontrado [33], podemos então definir a operação de *Max Pooling* matematicamente

como:

$$y_{kij} = \max_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} \tag{2.12}$$

onde x_{kij} será o valor retornado pela função de *pooling*, x_{kpq} é o elemento no ponto (p, q) na janela de *pooling* $\mathcal{R}_{ij}$, a Figura 10 representa um exemplo de aplicação da função *Max Pooling*.

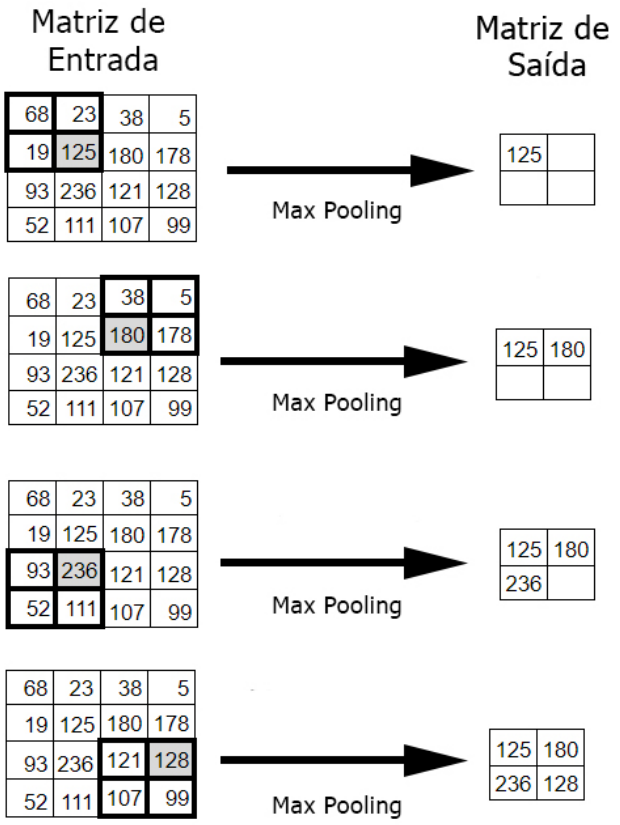

Figura 10 – *Max Pooling* em uma matriz de tamanho 4x4 com uma janela de *pooling* de 2x2 e *stride* de 2x2; Fonte: O Autor.

Analogamente podemos definir a função *Average Pooling*, porém agora o valor retornado pela função é a média aritmética dos elementos na janela de *pooling* [33]:

$$y_{kij} = \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} \tag{2.13}$$

sendo que $|\mathcal{R}_{ij}|$ representa o tamanho da região de *pooling*.

2.3.1.4 Camadas *Flatten* e *Dense*

Após a imagem passar pela camada de entrada e pelas transformações das camadas de convolução e *pooling* temos como resultante um *array* com n dimensões, porém para

se realizar a classificação é necessário se ter um *array* unidimensional ($n = 1$), está transformação ocorre na camada *flatten* preparando o *array* resultante para ser classificado.

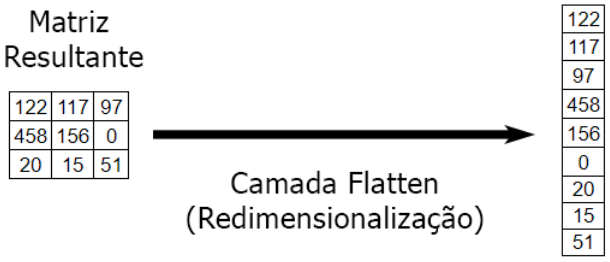

Figura 11 – Processo de redimensionalização do *array* durante a camada *flatten*; Fonte: O Autor.

Após o *array* ter sido redimensionado na camada *flatten* ele passa pela camada *dense* ou camadas totalmente conectadas, que contém neurônios dos quais estão diretamente conectados ao neurônios da camada anterior e seguinte, porém sem estarem conectados com os neurônios da mesma camada [20]. A camada *dense* de uma rede neural convolucional é análoga às redes neurais artificiais tradicionais (Figura 3), se fizermos um paralelo podemos afirmar que a camada de entrada das camadas totalmente conectadas seria os valores obtidos na redimensionalização do *array* na camada *flatten*.

2.3.2 Arquiteturas de Redes Neurais Convolucionais

Utilizando os conceitos apresentados na Seção 2.3.1 é possível construir diversas arquiteturas[5] de *CNNs* diferentes. Nessa secção será apresentado algumas arquiteturas já reconhecidas, seja pela sua eficiência ou contribuição para o desenvolvimento do campo de pesquisa.

Serão apresentadas as arquiteturas das redes *LeNet-5* que foi primeiramente proposta para a classificação de dígitos escritos manualmente em 1998, também será apresentada as arquiteturas das *CNNs AlexNet*, *VGG*, *Inception* e *ResNet*, que foram propostas para a competição ILSVRC, sendo essa uma competição que avalia algoritmos de detecção e classificação utilizando o conjunto de dados *ImageNet* que contém milhões de imagens rotuladas por humanos [34, 35].

2.3.2.1 *LeNet-5*

O modelo *LeNet-5* foi proposto inicialmente em 1998 por *Lecun et al.* para realizar a classificação de padrões de alta dimensão como caracteres escritos a mão sem a realização

[5] Arquitetura de rede neural é uma forma de designar a estrutura da rede neural, ou seja, de que forma suas camadas estão definidas.

do pré-processamento dos dados [8], já que realizar o tratamento de dados complexos como as imagens seria um trabalho árduo devido ao modo que o computador lê uma imagem. A arquitetura da rede contém sete camadas no total, destas cinco que contém parâmetros treináveis, além da camada de entrada [8], sendo assim a arquitetura da *LeNet-5* é mais simples se comparada a redes neurais convolucionais mais atuais. A descrição das camadas da arquitetura da *LeNet-5* pode ser consultada no Apêndice A.1 ou graficamente na na Figura 21.

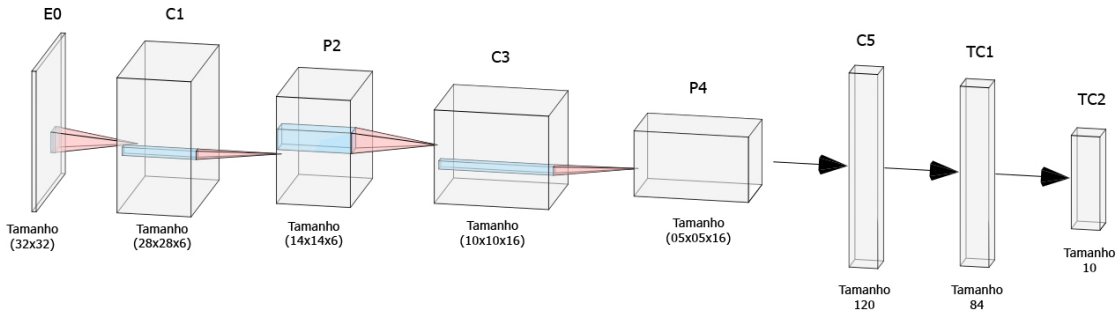

Figura 12 – Representação gráfica da arquitetura da rede *LeNet-5*; Fonte: O Autor.

Tomando como base a Figura 21 a camada E0 representa a camada de entrada da *LeNet-5* que recebe uma imagem com tamanho (32x32) com apenas um canal de cor, após a camada de entrada o *array* de entrada passa por uma camada de convolução (C1) com uma matriz de convolução de tamanho (5x5) e com 6 filtros de dimensionalidade. A camada P2 é uma camada de *pooling* que utiliza como função de *pooling* o *average pooling* com janela de *pooling* de (2x2). Na camada C3 ocorre a convolução com um *kernel* de tamanho (5x5) e 16 filtros de dimensionalidade. A camada P4 também é uma camada de *pooling*, com as mesmas configurações da camada P2. A camada C5 é a última camada de convolução com 120 filtros de dimensionalidade com uma matriz de convolução de tamanho (5x5), esta camada desempenha a função da camada *flatten* tradicional, transformando o *array* de tamanho (5x5x16) em um *array* de tamanho (120). Todas as camadas de convolução da *LeNet-5* tem como função de ativação a função *tanH* [8].

Após as camadas de convolução e *pooling* existem duas camadas totalmente conectadas a primeira com 84 neurônios e com função de ativação a função *tanh* e a camada de saída com 10 neurônios (quantidade de classes a ser classificada) e utilizando a função de ativação *softmax*.

2.3.2.2 *AlexNet*

A *AlexNet* proposta por *Krizhevsky, Sutskever & Hilton* em 2012 contém oito camadas com parâmetros treináveis, com um total de 60 milhões de parâmetros treináveis [9]. A camada de entrada da rede recebe uma imagem no tamanho (224x224x3), inicialmente é inserido uma camada de convolução, então é adicionado a primeira camada de *pooling* para diminuir a dimensionalidade do *array* a ser processado, após esta camada o *array* passa por uma segunda camada de convolução e então outra camada de *pooling* é adicionada, por fim o *array* resultante passa por uma sequência de três camadas de convolução.

Após todas as transformações ocorridas nas camadas de convolução e *pooling* a dimensionalidade do *array* é diminuída de forma que ele se torne unidimensional (vetor) na camada *flatten*, o último processo que ocorre na rede neural é a passagem do vetor a ser classificado nas camadas totalmente conectadas, a arquitetura *AlexNet* contém duas camadas *dense* ocultas e a camada de saída que retorna as probabilidades de pertencer do *array* pertencer às classes a serem classificadas. A descrição da arquitetura com os parâmetros utilizados na definição das camadas podem ser consultados no Apêndice A.2.

A *AlexNet* foi a primeira *CNN* a obter o melhor resultado na competição ILSVRC, onde em 2012 obteve uma porcentagem de erros de 15,3% [34], a arquitetura foi revolucionaria por ser a primeira a utilizar como função de ativação das camadas ocultas a função *ReLU* [9], por conseguir realizar a classificação de imagens coloridas e por utilizar as técnicas de *dropout* e *data augmentation*, sendo esses métodos de regularização (ver Seção 2.7), para diminuir as probabilidades de durante o treinamento da rede neural ocorrer um sobre-ajuste do modelo aos dados de treino e não conseguir realizar a classificação de imagens que nunca foram apresentadas a rede.

2.3.2.3 *VGG*

A arquitetura *VGG* (*Visual Geometry Group*) proposta por *Simonyan & Zisserman* em 2014 tem na camada de convolução matrizes de convolução apenas de tamanho (3x3) sendo esses menores comparados a outras arquiteturas disponível até o momento [10], existem duas versões principais da arquitetura de redes neuras *VGG*, uma contendo 19 camadas com parâmetros treináveis (*VGG-19*) e outra contendo 16 camadas com parâmetros treináveis (*VGG-16*) esta que obteve o segundo melhor resultado na competição ILSVRC-2014 com uma taxa de erro de $7,3\%$ [34].

A arquitetura *VGG* é uma evolução natural da *AlexNet* sendo mais profunda e com mais parâmetros que podem ser ajustados, contendo um total de 138 milhões de parâmetros treináveis para a *VGG-16* e 144 milhões de parâmetros treináveis para a *VGG-19*, a arquitetura e os parâmetros utilizados da *VGG-16* podem ser consultados no Apêndice A.3.

2.3.2.4 *Inception*

A arquitetura da rede neural convolucional *Inception*, ou *GoogLeNet*, tem como sua principal característica a presença do módulo *inception* que diminui o custo computacional do treinamento da rede neural [36], a ideia por trás da arquitetura é aumentar a profundidade da rede de modo que não seja custoso computacional, para isso nos módulos *inception* ocorrem convoluções que tem como tamanho de *kernel* (1x1), (3x3) ou (5x5) e uma camada de *pooling* com uma janela de *pooling* de tamanho (3x3), antes de todas as convoluções com tamanho (3x3) e (5x5) são inseridas convoluções com tamanho (1x1) [36], assim reduzindo assim o custo computacional e aumentando a profundidade da *CNN*.

Durante os módulos *inception* ao invés do *array* resultante da camada anterior passar apenas por uma camada de convolução ou *pooling* ele passa por camadas de convolução diferentes e camadas de *pooling* separadamente e os valores resultantes são concatenados no final do módulo por um filtro concatenador [37]. A arquitetura de rede *Inception* obteve o melhor resultado na competição ILSVRC-2014 para a detecção de objetos em imagens obtendo um taxa de erro de $6,67\%$.

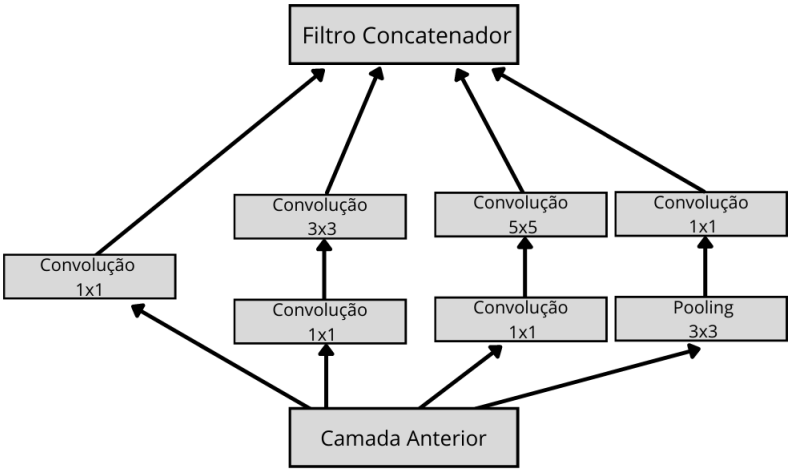

Figura 13 – Módulo *inception* 1; Adaptado de *Szegedy et al.* (2014).

Uma versão mais atualizada e mais otimizada do modelo *Inception* é a *CNN Inception-V3* proposto por *Szegedy et al.* em 2016, onde é proposto mais dois tipos de módulos *inception*, que podem ser consultados nas Figuras 14 e 15, a arquitetura proposta para a rede neural pode ser consultada no Apêndice A.4.

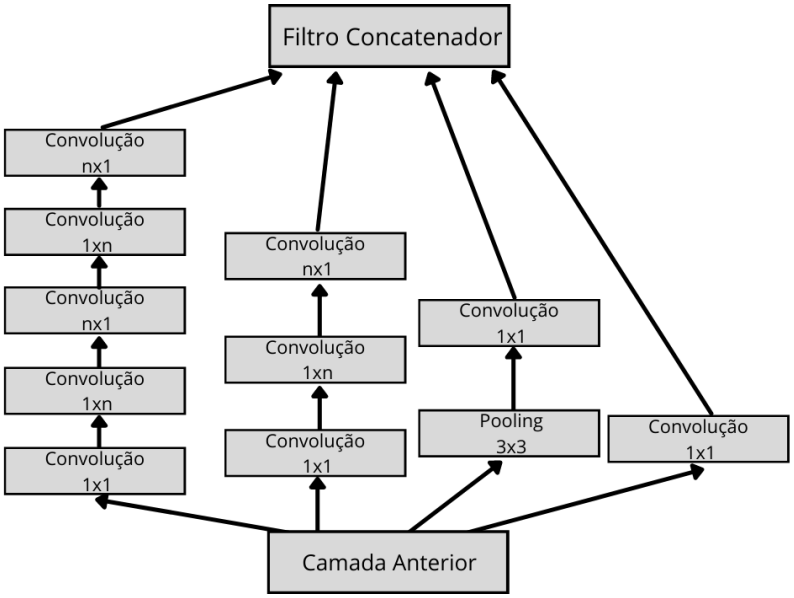

Figura 14 – Módulo *inception* 2; Adaptado de *Szegedy et al.* (2016).

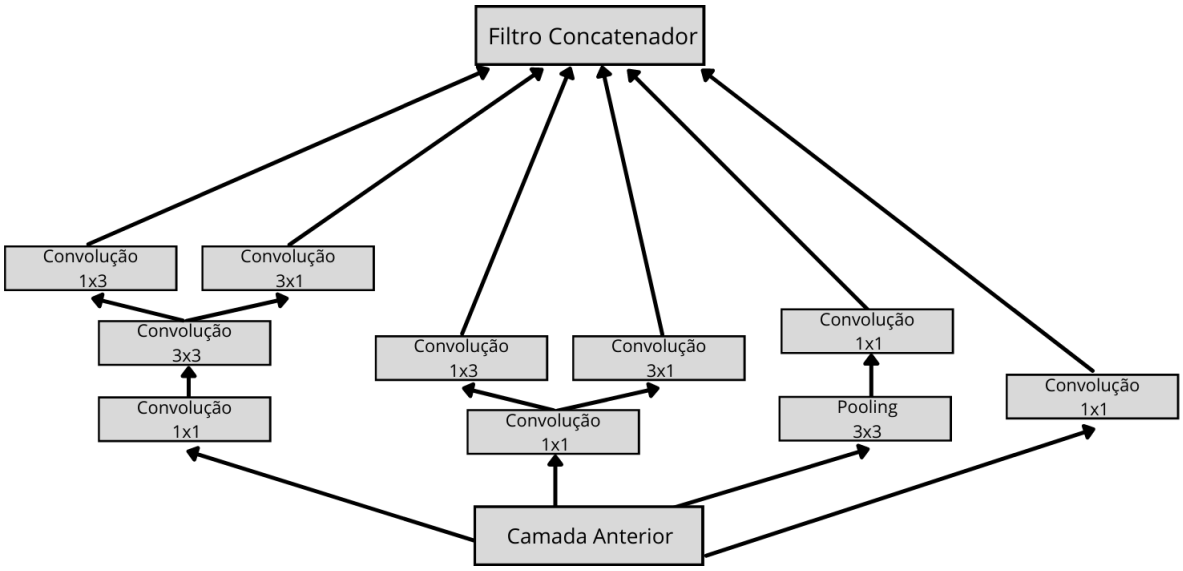

Figura 15 – Módulo *inception* 3; Adaptado de *Szegedy et al.* (2016).

2.3.2.5 *ResNet*

A arquitetura da rede neural convolucional *ResNet* (*Residual Network*) proposta por *He et al.* em 2015 obteve o melhor desempenho da competição ILSRVC-2015, desempenhando melhor que seres humanos. A rede neural é composta por blocos residuais que são mais fácil de serem otimizados [12], devido à presença da função identidade, sendo essa a função de mais fácil otimização, que ocorre nos "atalhos" dos blocos, representado pela Figura 16.

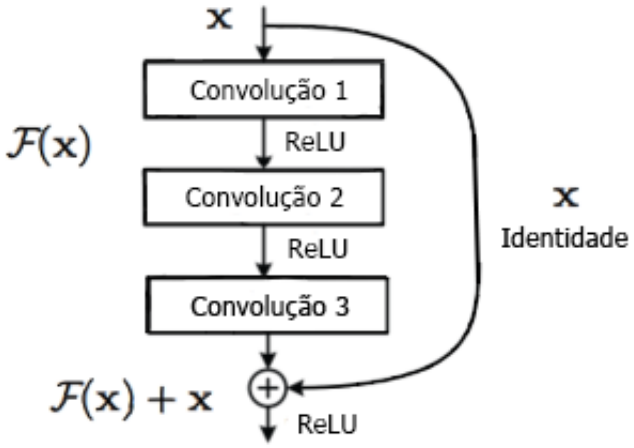

Figura 16 – Bloco residual de uma *ResNet-50*; Adaptado de *He et al.* (2016).

Se considerarmos uma função $\mathcal{H}(x)$ como sendo um mapeamento subjacente a ser otimizado em um bloco formado por algumas camadas da rede e x as entradas dessas camadas [12], podemos então definir a otimização em um bloco residual como sendo:

$$\mathcal{H}(x) := \mathcal{F}(x) + x \tag{2.14}$$

em *CNNs* anteriores, como os modelos *VGG* e *Inception* a atualização de $\mathcal{H}(x)$ se dava simplesmente por $\mathcal{F}(x)$.

Existem várias configurações diferentes para a *ResNet*, a mais comum e que será abordado nesse trabalho é a *ResNet-50*, que como o nome sugere é uma arquitetura que contem 50 camadas com parâmetros treináveis e contendo blocos residuais característicos da *ResNet*. A arquitetura da *ResNet-50* pode ser consultada no Apêndice A.5. Os atalhos representados pela Figura 16 acontecem nos blocos de convolução no Apêndice A.5 sempre a cada três convoluções.

2.4 Função de Custo

A função a qual queremos minimizar ou maximar é chamada de função objetivo[6], quando o objetivo é minimizar o valor da função objetivo a função é conhecida por função de custo, a escolha da função que vai ser utilizada como função de custo é de extrema importância para a obtenção dos melhores resultados possíveis durante o treinamento da rede neural [23, 25]. A função de custo é utilizada durante a fase de otimização dos parâmetros treináveis para o monitoramento do modelo, sendo aplicada nos dados de treinamento e de validação e junto com o valor de acurácia, é possível monitorar o processo de treinamento enquanto ele ocorre.

[6] Tradução livre para *Objective Function*

Para problemas de classificação de n classes é utilizada como função de custo a função *categorical cross entropy*, portanto neste caso se $\hat{y_1}, \ldots, \hat{y_n}$ são as probabilidades de n classes serem classificadas, utilizamos a função de ativação *softmax* dada pela Equação 2.6 e a classe real é a r-ésima classe [23], podemos definir a função de custo como:

$$\mathcal{L} = -log(soft(\hat{y}_r)) \tag{2.15}$$

onde $soft(\hat{y}_r)$ é a probabilidade dos valores passados para a função de ativação pertencerem a sua classe real, sendo assim caso a função *softmax* retorne o 1 como valor (100% de probabilidade de pertencerem à classe real r) a função de custo terá como valor 0, e caso a função *softmax* retorne como probabilidade 0 a função de custo vai tender a infinito.

2.5 Otimizadores

Ao se realizar o treinamento de uma rede neural estamos sempre procurando otimizar o vetor contendo os valores dos pesos sinápticos $\mathbf{W}$ tal que diminuam a função de custo [25], para garantir que o valor função de custo esteja diminuindo é utilizado algoritmos de otimização, ou otimizadores, que se baseiam no gradiente da função de custo para atualizar os pesos da rede neural [23], o algoritmo utilizado e que será estudado neste trabalho será o otimizador *Adam*.

O *Adam* (*Adaptive Moment Estimation*), proposto por *Kingma & Ba* em 2014 é um algoritmo com taxa de aprendizado adaptativo, sendo menos custoso computacionalmente comparado a outros algoritmos de otimização [25, 38]. Se considerarmos A_i como o valor médio exponencialmente do i-ésimo peso sinaptico w_i, este valor será atualizado com um parâmetro de decaimento $\rho \in (0, 1)$ [23], de modo que:

$$A_i \leftarrow \rho A_i + (1 - \rho) \left(\frac{\partial \mathcal{L}}{\partial \omega_i} \right)^2 \tag{2.16}$$

onde $\mathcal{L}$ é o valor da função de custo e ω_i é o i-ésimo peso sináptico. A Equação 2.16 é a equação utilizada no otimizador RMSP (*Root Mean Square Propagation*).

Simultaneamente um valor exponencialmente suavizado do gradiente é mantido para o qual o i-ésimo é denotado F_i, porém esta realizada como um parâmetro de decaimento diferente denominado ρ_f:

$$F_i \leftarrow \rho_f F_i + (1 - \rho_f) \left(\frac{\partial \mathcal{L}}{\partial \omega_i} \right) \tag{2.17}$$

a Equação 2.17 é a equação utilizada no algoritmo de otimização de parâmetros *Momentum*. Com as Equações 2.16 e 2.17 é possível definir o modo em que novos valores dos parâmetros

serão otimizados no decorrer do treinamento do modelo:

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} F_i \tag{2.18}$$

onde α é uma constante de ajuste denominada taxa de aprendizado, sendo esta definida na construção da rede neural.

2.6 Métricas de Avaliação

Nessa seção será apresentada a definição de algumas métricas para realizar a avaliação do modelo classificatório, as métricas que serão estudadas e utilizadas serão a acurácia, função de custo, precisão, sensibilidade e F_1-*Score*.

2.6.1 Matriz de Confusão

A partir da classificação realizadas pelo modelo classificatório apos o treinamento é possível realizar a construção da matriz de confusão, onde cada elemento e_{ij} da matriz de confusão fornece o número de exemplos, cuja verdadeira classe era C_i e o foi classificado como C_j [39]. Teremos que a diagonal principal da matriz de confusão como sendo o acerto do modelo classificatório e os valores fora da diagonal principal como sendo os erros cometidos pelo modelo, podemos então denominar para um problema de classificação binária o elemento e_{11} como sendo o valor de verdadeiro positivo (VP), e_{22} o número de verdadeiro negativo (VN), e_{12} o número de falso negativo (FN) e e_{21} o número de falsos positivo (FP), sendo assim possível construir a Tabela 1.

Tabela 1 – Matriz de confusão para um problema de classificação binária.

	Condição Positiva	Condição Negativa
Condição Positiva Prevista	Verdadeiro Positivo (VP)	Falso Positivo (FP)
Condição Negativa Prevista	Falso Negativo (FN)	Verdadeiro Negativo (VN)

Os valores de VP que são os números das classificações positivas que realmente são positivas, VN representam a quantidade classificações negativas que são realmente negativas, FN são as classificações realizadas como negativas, porém com o dado pertencente a classe positiva e FP são o número de classificações realizadas erroneamente como positivo, porém pertencendo a classe negativa [40].

2.6.2 Acurácia

A acurácia ou taxa de acerto do modelo classificatório é a razão entre o número de amostras classificadas corretamente e o número total de amostras do conjunto de dados [40].

O valor de acurácia varia de 1,0, que é o melhor resultado possível para a acurácia (o modelo acertou todas as classificações) e 0,0 sendo o pior resultado para a acurácia (o modelo não acertou nenhuma classificação). Utilizando os valores de classificações corretas e erradas realizadas pelo modelo que podem ser obtidos na matriz de confusão, podemos definir a acurácia como os acertos do modelo divido pelo total de amostras:

$$acc = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.19}$$

Porém para problemas de classificações com classes de dados desbalanceados a acurácia pode não nos fornecer adequadamente a informação sobre o modelo em relação a um dado grupo específico [39]. Se considerarmos, por exemplo, um conjunto de dados com duas classes onde a classe com menor quantidade de dados é representada por 1% dos dados, se o modelo classificatório treinado, classificar todos os dados como sendo da classe que contem a maior quantidade de dados a taxa de acerto resultante será de 99%, tornando assim o classificador inútil caso o objetivo seja encontrar os dados da classe minoritária.

2.6.3 Precisão, Sensibilidade e F_1-*score*

A sensibilidade, também conhecida como *recall* pode ser definida como a taxa de verdadeiro positivos e retorna a razão entre os números de exemplos positivos corretamente classificados e o número total de exemplos positivos originais [39, 41], podendo ser definida pela seguinte equação:

$$s = \frac{VP}{VP + FN} \tag{2.20}$$

onde VP e FN são os valores de verdadeiro positivo e falso negativo obtido na Tabela 1. Analogamente podemos definir a taxa de verdadeiros negativos, ou especificidade como:

$$e = \frac{VN}{VN + FP} \tag{2.21}$$

Em contrapartida, a precisão nos corresponde à razão entre o número de exemplos positivos corretamente classificados e o número total de exemplos identificados como positivos pelo modelo [39]:

$$p = \frac{VP}{VP + FP} \tag{2.22}$$

Utilizando os valores de sensibilidade e precisão é possível tomar a média harmônica entre as Equações 2.20 e 2.22 e obter outra métrica de avaliação, o *F-score* [41]:

$$F_\beta = (1 + \beta^2)\frac{s \cdot p}{s + \beta^2 p}$$

$$F_\beta = \frac{(1+\beta^2)VP}{(1+\beta^2)VP + \beta^2 FN + FP} \tag{2.23}$$

Assumindo $\beta = 1$ obtemos o F_1-*score*:

$$F_1 = \frac{2VP}{2VP + FN + FP} = 2 \cdot \frac{p \cdot s}{p + s} \tag{2.24}$$

2.7 Métricas de Regularização

Um problema que pode acontecer após o treinamento do modelo é o sobre ajuste[7] do modelo classificatório que ocorre quando existe uma grande diferença entre o erro para os dados de treinamento e o erro para os dados de teste ou validação [25]. Para que diminuir as possibilidades da ocorrência do sobre ajuste existem as técnicas de regularização, sendo elas estratégias usadas para reduzir o erro nos dados de teste, porém possivelmente aumentando o erro de treinamento adicionando uma penalidade para a função de custo usada [23, 25], tornando assim diminuindo a diferença entre o erro de treinamento e teste. As técnicas de regularização tratadas nessa seção serão o *data augmentation* e o *dropout*.

2.7.1 *Data Augmentation*

O método mais fácil e comum para reduzir o sobre ajuste para modelos de classificação de imagens é aumentar artificialmente o conjunto de dados [9], para conseguir aumentar os dados a partir dos dados existentes é utilizado o *data augmentation*, que são simples distorções aplicadas nas imagens como translações, rotações e alteração no brilho da imagem, [42] gerando assim, imagens novas e diferentes a partir das contidas no conjunto de dados original, essas novas imagens geradas aleatoriamente serão utilizadas para o treinamento da rede neural convolucional.

Algumas distorções que podem ser aplicadas nas imagens contidas no conjunto de dados original são: a inversão da imagem vertical ou horizontalmente, a rotação para a esquerda ou para a direita da imagem no seu próprio eixo entre 1^o e 365^o e a alteração na escala da imagem, onde é aplicado um *zoom* aleatório dentro de uma escala passada para a função [43, 44].

2.7.2 *Dropout*

Outra técnica de regularização que pode diminuir as possibilidades de ocorrer o sobre ajuste do modelo classificatório durante o treinamento é a adição do *dropout* na arquitetura da *CNN*. Como o nome sugere alguns neurônios são removidos temporariamente

[7] Tradução livre para *overfitting*

da rede neural junto com suas conexões de maneira aleatória [45], porém esta técnica só é aplicada durante o processo de treinamento quando o modelo realizar uma classificação apos o treinamento os neurônios são ativados novamente.

Além de diminuir as possibilidades de sobre ajuste do modelo o uso do *dropout* reduz significativamente o custo computacional do treinamento do modelo, porém a quantidade de neurônios removidas depende do tamanho da camada em que a técnica de regularização é aplicada e também da taxa de desligamento aleatório utilizada na definição da camada pode reduzir a capacidade efetiva de um modelo [25, 45], dependendo da quantidade de neurônios desligado.

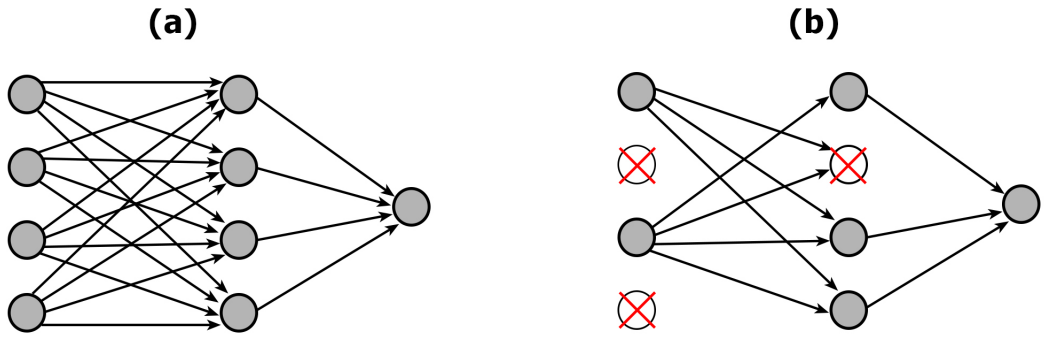

Figura 17 – (a) RNA com uma camada oculta; (b) *Dropout* de 50% aplicado na camada de entrada na RNA da imagem (a) e 25% na camada oculta; Adaptado de *Srivastava, Krizhevsky, Sutskever* e *Salakhutdinov* (2014).

3 Materiais e Métodos

3.1 Implementação do Código

Para implementar o problema computacionalmente, foi utilizado a linguagem de programação *Python* no ambiente *Jupyter Notebook* na nuvem *Google Colaboratory*, que fornece uma GPU NVIDIA Tesla K80 12GB de maneira gratuita, além de permitir a integração de maneira simplificada com o *Google Drive*, sendo este o ambiente onde estão armazenadas as imagens que serão utilizadas para a otimização, validação e para o teste do modelo.

Para se realizar a construção e o treinamento do modelo foi utilizado a biblioteca de aprendizado profundo *Keras* que facilita a construção do modelo já que a biblioteca disponibiliza as funções utilizadas para a definição das camadas da rede neural. Para e gerar as métricas de avaliação foi utiliza a biblioteca de *Sckit-Learn*, que já contém todas as métricas utilizadas.

3.2 Dados

Os dados utilizados para a realização do trabalho foram retirados do conjunto de dados aberto *"Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images"*, que contem milhares de imagens já rotuladas e validades de exames de radiografia do tórax [7], todas imagens estão todas no formato JPEG e estão previamente dividas em duas pastas, uma com dados reservados para o processo de treinamento do modelo classificatório e a outra com imagens reservadas para se realizar o teste do modelo.

A pasta com dados para o treinamento contêm 5082 imagens no total, das quais estão subdivididas em outras duas pastas, uma contendo 1299 imagens de pacientes que não contém nenhuma doença, e a outra com 3783 imagens de radiografias do tórax diagnosticadas com algum tipo de pneumonia. A pasta com dados reservados ao teste do modelo contém um total de 624 imagens e esta subdivida igualmente, porém contendo 234 imagens de pulmões diagnosticados como saudáveis e 390 imagens de pulmões com algum tipo de pneumonia.

Ao se analisar os dados foi possível perceber que as imagens rotuladas como pneumonia estão ainda rotuladas como pneumonia viral e bacteriana, porém estando nas mesmas pastas, sendo assim a pasta de imagens contendo imagens com pneumonia foram separadas em duas uma contendo imagens rotuladas como pneumonia viral e outra como pneumonia bacteriana. O esquema das relações das pastas dos dados podem ser

consultadas na Figura 18.

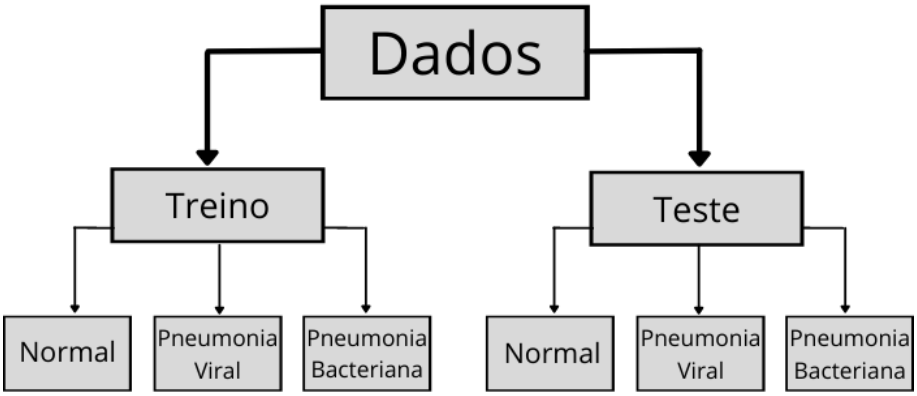

Figura 18 – Relação das pastas dos dados; Fonte: O Autor.

Foram também retirados alguns dados da pasta de imagens reservadas para o treinamento do modelo visando se criar uma pasta contendo dados para a validação do modelo, ou seja, dados reservados para se realizar o teste do modelo durante o processo de treinamento. O novo esquema de pastas dos dados podem ser consultados na Figura 19.

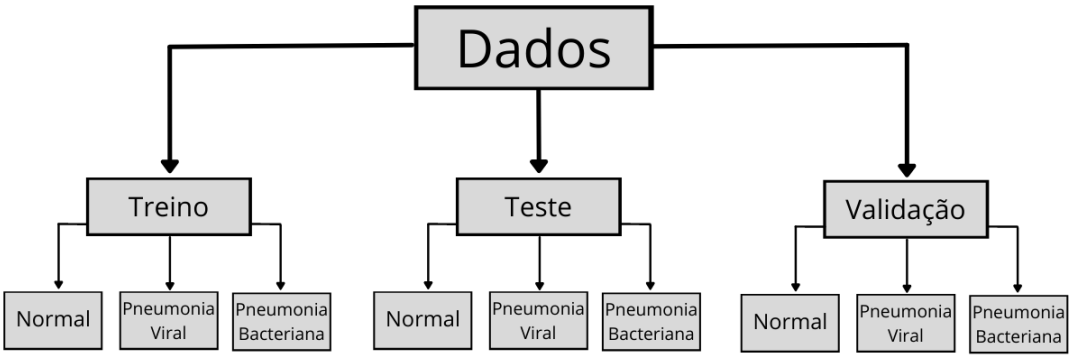

Figura 19 – Relação das pastas dos dados após a criação dos dados de validação; Fonte: O Autor.

O total de imagens em cada pasta após as modificações pode ser consultada na Tabela 2.

Tabela 2 – Quantidade de imagens pertencente a cada classe.

	Normal	**Pneumonia Bacteriana**	**Pneumonia Viral**
Treinamento	1299	2488	1295
Teste	234	242	148
Validação	50	50	50

3.2.1 Pesando os Dados

Como é possível notar na Tabela 2 algumas classes das imagens reservadas ao treinamento do modelo contem significativamente mais imagens que outras, apesar de ser possível realizar o treinamento de uma *CNN* com as classes de dados desbalanceadas não é aconselhável, já que as classes que contém menor quantidade de dados podem sofrer com baixa taxa de acerto. Para contornar este problema a cada classe dos dados de treino foram aplicados pesos as imagens, ou seja, uma imagem que pertence a uma classe que contem menos dados terá maior relevância no processo de treinamento do modelo classificatório.

Considerando que as imagens da classe com maior quantidade de imagens (Pneumonia Bacteriana) como tendo peso 1, podemos definir que as imagens da classe Pneumonia Viral terão peso 1,921 e as imagens de radiografia de tórax saudáveis terão peso de 1,915.

3.3 Parâmetros e Valores para o *Data Augmentation*

Durante a fase de treinamento do modelo foi se utilizado do *data augmentation* para diminuir as probabilidades de ocorrer o sobre ajuste do modelo durante a sua otimização. As distorções utilizadas nas imagens foram a alteração da escala da imagem (*zoom*), caso o parâmetro utilizado para alteração na escala seja maior que 1.0 a nova imagem sera uma ampliação da original, e caso seja menor que 1.0 a nova imagem será a original com uma escala menor, outra transformação utilizada foi a rotação das imagens, as novas imagens serão geradas a partir de uma rotação no próprio eixo para direita ou esquerda tendo como ângulo de rotação máximo o valor passado como parâmetro [43], as imagens foram também deslocadas horizontal e verticalmente e também foram invertidas no eixo horizontal de maneira aleatória.

Ao se realizar as transforações as novas imagens geradas podem ter áreas sem preenchimento de cor, para isso as novas áreas foram preenchidas com a cor preta de maneira constante, já que é a cor de fundo das imagens. As distorções utilizadas e seus valores podem ser consultadas na Tabela 3.

Tabela 3 – Parâmetros e valores utilizados durante o *data augmentation*.

Distorções	**Valores**
Faixa de Alteração na Escala *(Zoom Range)*	0,9 a 1,25
Faixa de Rotação *(Rotation Range)*	10^o
Faixa de Deslocamento Horizontal *(Width Shift Range)*	0,1
Faixa de Deslocamento Vertical *(Height Shift Range)*	0,1
Modo de Preenchimento *(Fill Mode)*	Constante
Cval	0

Alguns exemplos de novas imagens geradas a partir das junções das transformações

do *data augmentation* podem ser consultadas na Figura 20, sendo essas as imagens que serão utilizadas durante o treinamento, aumentando assim a capacidade de generalização do modelo [9].

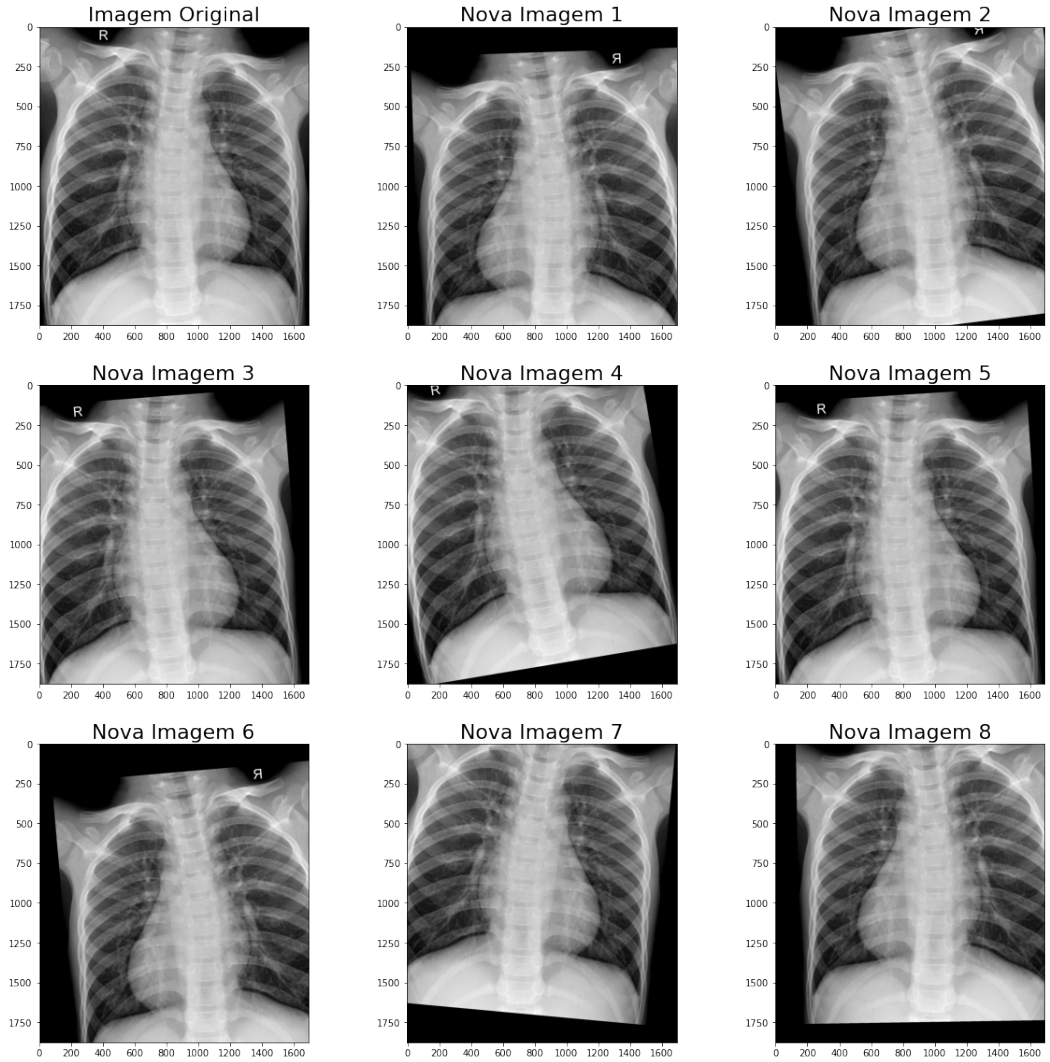

Figura 20 – Exemplo de imagens geradas a partir dos valores utilizados para as distorções aplicadas a uma imagem durante o *data augmentation*; Fonte: O Autor.

.

3.4 Otimização dos Hiper-Parâmetros

Os hiper-parâmetros de uma rede neural são os valores que são escolhidos na hora da construção da rede neural e são definidos manualmente e interferem diretamente no desempenho da rede neural [46], porém existem alguns algorítimos que permitem a otimização destes hiper-parâmetros, como o *Hyperband* que foi utilizado neste trabalho.

Utilizando o *Hyperband* foi possível encontrar os valores de taxa de aprendizagem dos otimizadores das redes neurais e o número de neurônios das camadas totalmente conectadas

que realizam as melhores classificações para o conjunto de dados de treinamento. Os valores de hiper-parâmetros utilizados podem ser consultados na Tabela 4.

Tabela 4 – Melhores valores encontrados para o número de neurônios da camada totalmente conectada e a taxa de aprendizado utilizada nos otimizadores.

Arquitetura de CNN	**Número de Neurônios**	**Taxa de Aprendizado**
LeNet-5	430	1×10^{-4}
AlexNet	840	1×10^{-5}
VGG-16	840	1×10^{-5}
ResNet-50	-	1×10^{-5}
Inception V3	-	1×10^{-5}

Os valores das taxas de aprendizado foram encontrados dentro os seguintes valores $[1 \times 10^{-1}; 1 \times 10^{-2}; 1 \times 10^{-3}; 1 \times 10^{-4}; 1 \times 10^{-5}]$. Como as arquiteturas de redes neurais convolucionais *LeNet-5*, *AlexNet* e *VGG-16* contém mais de uma camada totalmente conectada, é possível encontrar o melhor número de neurônio para se realizar a classificação das imagens.

Para a *CNN LeNet-5* o intervalo utilizado para encontrar o número de neurônios foi de 10 a 1000 neurônios, sendo o melhor valor encontrado para se realizar a classificação de 430 neurônios, para as *CNNs AlexNet* e *VGG-16* o intervalo de busca para o número de neurônios foi de 100 a 5000 e em ambas o número de neurônios para as camadas *dense* foi de 840 neurônios.

4 Resultados

A partir da metodologia descrita foi possível realizar o treinamento das diferentes arquiteturas de redes neurais convolucionais para o problema de classificação das imagens resultantes de radiografia do tórax por um total de 100 épocas cada. Inicialmente as *CNNs* foram treinadas para conseguir diferir imagens de pulmões saudáveis, com pneumonia viral e com pneumonia bacteriana (classificação multi-classe), sendo assim o tamanho da camada de saída das arquiteturas de redes neurais convolucionais foram alterados para terem três neurônios na camada de saída, sendo este o número de classes a serem classificadas.

Com as redes neurais convolucionais treinadas para classificar as três classes foi possível também realizar a classificação binária das imagens, ou seja, realizar a classificação se a imagem passada a *CNN* era originalmente rotulada como radiografia de tórax de um pulmão saudável ou continha algum tipo de pneumonia.

4.1 LeNet-5

Utilizando a arquitetura da rede neural convolucional *LeNet-5*, porém com a camada *dense* modificada para ter 430 neurônios e utilizando como taxa de aprendizado para o otimizador *Adam* 1×10^{-4}, tanto o número de neurônios utilizados para a camada totalmente conectada e a taxa de aprendizado foram encostradas a partir da otimização de hiper-parâmetros, sendo assim foi obtido um modelo com um total de 104.015 parâmetros que podem ser ajustados durante o treinamento do modelo, foi possível realizar a otimização dos parâmetros ajustáveis com os dados reservados ao treinamento do modelo pelo processo de treinamento por épocas, o modelo classificatório foi treinado por um total de 100 épocas, os gráficos gerados durante o treinamento podem ser consultados na Figura 21.

Após o processo de treinamento do modelo classificatório foi realizado a classificação dos dados já rotulados reservados ao teste do mesmo, a partir das classificações realizadas foi obtido uma taxa de acerto de 79,80%, além da acurácia foi possível, com os dados utilizados para a avaliação do modelo construir a matriz de confusão contendo os erros e os acertos do modelo para cada classe, esta podendo ser consultada na Tabela 5.

Com os dados de verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo que foram retirados da matriz de confusão, foi possível obter os valores das outra métricas de avaliação utilizadas para validar o modelo, estas podendo ser consultadas na Tabela 6.

Ao analisar a matriz confusão gerada para os dados reservados ao teste do modelo é possível perceber que o modelo obteve baixa taxa de acerto para imagens rotuladas

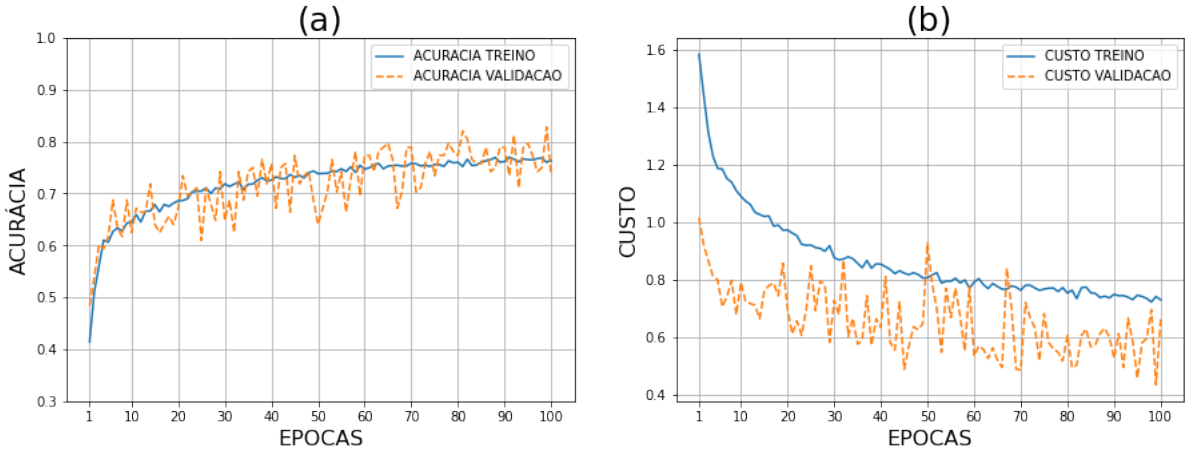

Figura 21 – Gráfico do treinamento do modelo *LeNet-5* por 100 épocas para **(a)** Acurácia para os dados de treino (linha azul) e de validação (linha laranja) e **(b)** Valor da função de custo para os dados de treino (linha azul) e de validação (linha laranja).

Tabela 5 – Matriz de confusão para o modelo *LeNet-5*.

	Pneumonia Bacteriana	Normal	Pneumonia Viral
Pneumonia Bacteriana	213	14	10
Normal	05	224	05
Pneumonia Viral	34	53	61

Tabela 6 – Métricas de avaliação para o modelo *LeNet-5*.

	Sensibilidade	Precisão	F1-Score
Normal	0,957	0,757	0,845
Pneumonia Bacteriana	0,880	0,845	0,862
Pneumonia Viral	0,412	0,803	0,545

como pneumonia viral, já que de 148 imagens apenas 61 foram classificadas corretamente, além de 34 classificadas como pneumonia bacteriana e 53 que foram classificadas como radiografia de pulmões saudáveis, sendo assim, este é o motivo da baixa sensibilidade e F_1-*Score* da pneumonia bacteriana, como possível verificar na Tabela 6.

Para o caso de classificar imagens em radiografias do tórax de pulmões saudáveis ou contendo algum tipo de pneumonia a *CNN* baseada na arquitetura da *LeNet-5* obteve uma acurácia de 86,85% para os dados reservados ao teste do modelo. Assim como para o caso de classificação multi-classe foi possível construir a matriz de confusão para as classificações realizadas corretamente e as realizadas erroneamente:

Tabela 7 – Matriz de confusão para o problema binário do modelo *LeNet-5*.

	Normal	Pneumonia
Normal	224	10
Pneumonia	72	318

Com os dados retirados da matriz de confusão para a classificação binária é possível também encontrarmos os valores das métricas de avaliação do modelo classificatório para esta situação:

Tabela 8 – Métricas de avaliação para o problema de classificação binário do modelo *LeNet-5*.

	Sensibilidade	Precisão	F1-Score
Normal	0,957	0,757	0,845
Pneumonia	0,815	0,970	0,886

Ao se analisar agora as métricas de avaliação do modelo e a matriz de confusão para o caso de classificação binária é possível notar que o modelo obteve métricas razoáveis, já que as duas classes obtiveram mais de 0,84 de F_1-*Score*, levando em consideração a simplicidade da arquitetura utilizada para realizar a classificações das imagens.

4.2 AlexNet

Para o modelo classificatório baseado na arquitetura de rede neural convolucional *AlexNet* o número de neurônios utilizados na primeira e na segunda camada *dense* foram modificados para 840, além da taxa de aprendizado utilizado para o otimizador *Adam* sendo esta de 1×10^{-5}, obtendo assim uma rede neural com total de 9.809.771 parâmetros treináveis. A rede neural convolucional foi treinada por um total de 100 épocas utilizando os dados reservados para o treinamento do modelo com seus devidos pesos. O gráfico do treinamento do modelo pode ser consultado na Figura 22.

Ao se analisar o gráfico apresentado na Figura 22 é possível perceber que o modelo conseguiu ajustar bem os parâmetros durante o processo de treinamento conseguindo também realizar a classificação de imagens que não foram utilizadas no processo de otimização da rede neural com bom desempenho, já que é possível notar que tanto o gráfico da acurácia quanto no gráfico para o valor da função de custo as curvas que representam os dados de treino e validação apresentam a mesma tendencia.

Após concluído a fase de treinamento do modelo foi possível obter a taxa de acerto do modelo a partir da classificação dos dados reservados ao teste já que estes estão previamente rotulados, portanto a acurácia obtida foi de $87,01\%$. Com as classificações

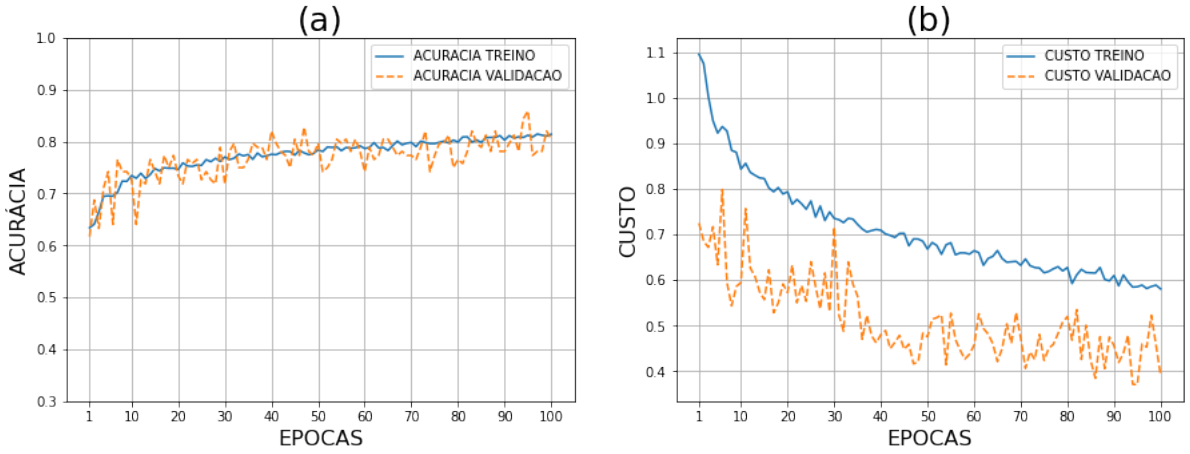

Figura 22 – Gráfico do treinamento do modelo *AlexNet* por 100 épocas para **(a)** Acurácia para os dados de treino (linha azul) e de validação (linha laranja) e **(b)** Valor da função de custo para os dados de treino (linha azul) e de validação (linha laranja).

realizadas pelo modelo treinado foi possível também construir a matriz de confusão para os dados reservados ao teste do modelo, esta podendo ser conferida na Tabela 9.

Tabela 9 – Matriz de confusão para o modelo *AlexNet*.

	Pneumonia Bacteriana	Normal	Pneumonia Viral
Pneumonia Bacteriana	223	04	15
Normal	02	190	42
Pneumonia Viral	18	0	130

Com os dados da matriz de confusão é possível também encontrar outras métricas realizar a avaliação da *AlexNet*, estas podendo ser consultadas na Tabela 10.

Tabela 10 – Métricas de avaliação para o modelo *AlexNet*.

	Sensibilidade	Precisão	F1-Score
Normal	0,812	0,979	0,888
Pneumonia Bacteriana	0,921	0,918	0,920
Pneumonia Viral	0,878	0,695	0,776

Pelas métricas apresentadas na Tabela 10 foi possível notar que o modelo obteve boa performance ao se realizar as classificações de imagens de radiografia do tórax rotuladas como Pneumonia Bacteriana, já que obteve como F_1-*Score* 0,920, porém o modelo não obteve uma bons resultados na classificação de imagens rotuladas como Pneumonia Viral,

obtendo como F_1-*Score* 0,776, sendo o motivo do mau desempenho do modelo para a classe a baixa precisão apresentada para os dados de teste, já que o mesmo obteve um número elevado de falsos positivos quando comparado a outras classes.

Agora, para o caso da classificação binária a taxa de acerto obtida pelo modelo foi de 92,31% para os dados de teste do modelo, utilizando também os dados de teste foi possível construir a matriz de confusão:

Tabela 11 – Matriz de confusão para o problema binário do modelo *AlexNet*.

	Normal	Pneumonia
Normal	190	44
Pneumonia	04	386

Com a matriz confusão é possível agora encontrar as métricas de avaliação para o caso de classificação binário, esta podendo ser consultada na Tabela 12:

Tabela 12 – Métricas de avaliação para o problema de classificação binário do modelo *AlexNet*.

	Sensibilidade	Precisão	F1-Score
Normal	0,812	0,979	0,888
Pneumonia	0,990	0,898	0,941

Analisando a acurácia e as outras métricas utilizadas para a avaliação do modelo treinado é possível notar o bom desempenho para o caso da classificação de pneumonia ou não nas imagens de teste, assim como para o caso multi-classe, levando em consideração a simplicidade do modelo utilizado.

4.3 VGG-16

A partir da arquitetura de rede neural *VGG-16* foi proposta um modelo classificatório com o mesmo número de camadas, porém o número de neurônios das camadas totalmente conectadas e a taxa de aprendizado foram alteradas, os valores encontrados a partir da otimização de hiper-parâmetros foram de 840 neurônios nas duas camadas *dense* e como valor de taxa de aprendizado para o otimizador *Adam* 1×10^{-5}. Com isso o modelo classificatório obtido contém um total de 36.497.259 parâmetros treináveis e foi treinado para a classificação das três classes por 100 épocas, o gráfico do desenvolvimento do modelo durante o treinamento pode ser consultado na Figura 23.

Utilizando o modelo já treinado para realizar a classificação das imagens já rotuladas do conjunto de dados reservados ao teste do modelo, foi então, obtida taxa de acerto de 87,02% para estas imagens. A partir destas classificações foi possível também construir

a matriz de confusão, podendo ser consultada na Tabela 13, sendo assim possível ver os erros e acertos do modelo para cada classe.

A partir da matriz de confusão construída para o modelo baseado na arquitetura *VGG-16* é possível se obter as outras métricas para a avaliação do desempenho do modelo (Tabela 14) além da taxa de acerto.

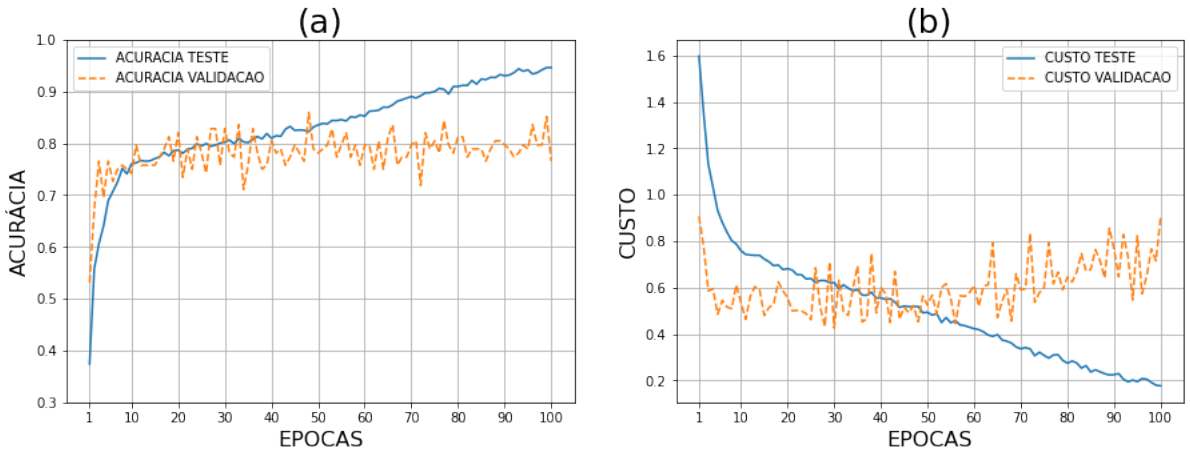

Figura 23 – Gráfico do treinamento do modelo VGG-16 por 100 épocas para **(a)** Acurácia para os dados de treino (linha azul) e de validação (linha laranja) e **(b)** Valor da função de custo para os dados de treino (linha azul) e de validação (linha laranja).

Ao analisar os gráficos da acurácia e do valor da função de custo ao decorrer das épocas de treinamento do modelo, é possível notar que aproximadamente após a época 40 o modelo classificatório começou a apresentar um sobre-ajuste dos parâmetros, já que apesar de a acurácia e o valor da função de custo para os dados de validação terem atingido um platô, o modelo ainda estava conseguindo se otimizar para os dados reservado para o treinamento, já que a função de custo ainda continuo em tendência de queda e a acurácia em tendência de alta. Um dos motivos pelo qual pode ter ocorrido o sobre-ajuste é o alto número de parâmetros treináveis do modelo.

Tabela 13 – Matriz de confusão para o modelo *VGG-16*.

	Pneumonia Bacteriana	Normal	Pneumonia Viral
Pneumonia Bacteriana	207	10	25
Normal	01	221	12
Pneumonia Viral	30	03	115

Tabela 14 – Métricas de avaliação para o modelo VGG-16.

	Sensibilidade	Precisão	F1-Score
Normal	0,944	0,944	0,944
Pneumonia Bacteriana	0,855	0,870	0,863
Pneumonia Viral	0,777	0,757	0,767

Agora, para o caso da classificação de apenas duas classe o modelo obteve uma taxa de acerto para os dados de teste de 95,83%, os erros e os acertos das classificações podem ser consultadas na Tabela 15.

Tabela 15 – Matriz de confusão para o problema binário do modelo VGG-16.

	Normal	Pneumonia
Normal	221	13
Pneumonia	13	377

Utilizando o número de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, retirados da matriz de confusão é possível encontrar as métricas de avaliação do modelo para a classificação de duas classes.

Tabela 16 – Métricas de avaliação para o problema de classificação binário do modelo VGG-16.

	Sensibilidade	Precisão	F1-Score
Normal	0,944	0,944	0,944
Pneumonia	0,967	0,967	0,967

Apesar do sobre-ajuste do modelo durante a fase de treinamento o modelo classificatório baseado na arquitetura *VGG-16* conseguiu performar bem tanto para o caso de classificação multi-classe como para o caso de classificação binário, tendo em vista as acurácias obtidas para ambos os casos (87,01% para o caso multi-classe e 92,31% para a classificação binária) e os valores das métricas de avaliação utilizadas para os dois casos.

4.4 Inception V3

Utilizando da arquitetura *Inception V3* foi construído o modelo classificatório capaz de realizar a classificação de uma radiografia de tórax de um pulmão saudável, com pneumonia viral e com pneumonia bacteriana. Utilizando a técnica de otimização de hiperparâmetros foi possível encontrar a melhor taxa de aprendizado para o otimizador, sendo esta 1×10^{-5}, o modelo final contém um total de 22.955.811 parâmetros, destes 22.921.379

parâmetros que serão ajustados durante a fase de treinamento e 34.432 parâmetros não treináveis.

O modelo baseado na arquitetura *Inception V3* foi treinado por um total de 100 épocas, onde o gráfico da acurácia e o valor da função de custo do modelo podem ser consultados na Figura 24. Com os parâmetros dos modelos já otimizado foi possível realizar a classificação dos dados reservados ao teste do modelo, assim obtendo uma taxa de acerto de 84,46%.

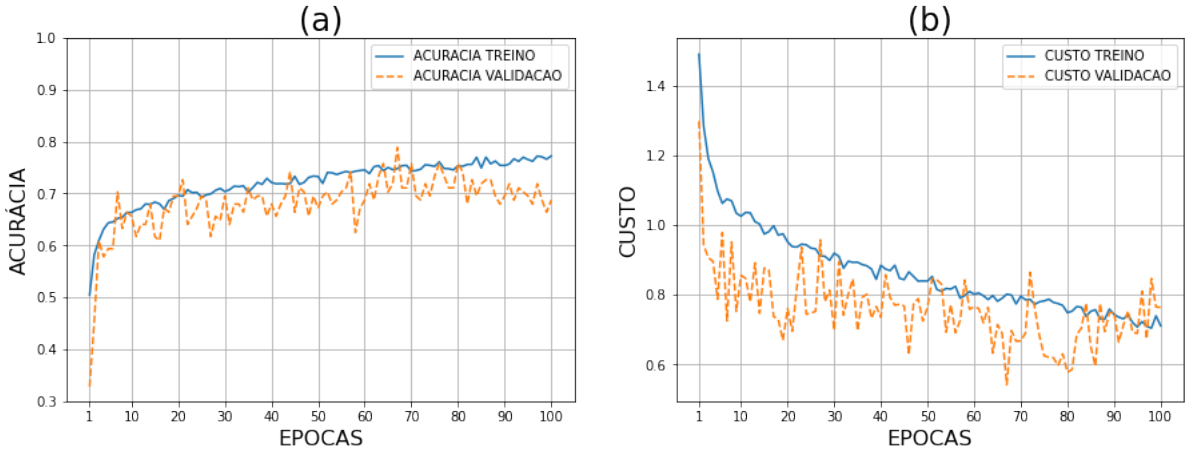

Figura 24 – Gráfico do treinamento do modelo *Incpetion V3* por 100 épocas para **(a)** Acurácia para os dados de treino (linha azul) e de validação (linha laranja) e **(b)** Valor da função de custo para os dados de treino (linha azul) e de validação (linha laranja).

Ao analisar os gráficos da acurácia e do valor da função de custo é possível ver que o modelo se ajustou bem durante a fase de treinamento, conseguindo se otimizar com o decorrer das épocas tanto para os dados de treinamento quanto para os dados de validação não apresentando sobre ajuste ou sob ajuste aparente. Com as classificações realizadas dos dados reservados ao teste do modelo foi possível também construir a matriz de confusão com os erros e acertos para cada classe, podendo ser consultadas na Tabela 17.

Tabela 17 – Matriz de confusão para o modelo *Inception V3*.

	Pneumonia Bacteriana	Normal	Pneumonia Viral
Pneumonia Bacteriana	215	19	08
Normal	10	213	11
Pneumonia Viral	32	17	99

Com os valores obtidos na matriz de confusão para a classificação multi classe do modelo foi também possível encontrar as métricas de avaliação para os dados de teste.

Tabela 18 – Métricas de avaliação para o modelo *Inception V3*.

	Sensibilidade	Precisão	F1-Score
Normal	0,910	0,855	0,882
Pneumonia Bacteriana	0,888	0,837	0,862
Pneumonia Viral	0,669	0,839	0,744

O modelo obteve boas métricas de avaliação considerando os valores obtidos na matriz de confusão e as outras métricas obtidas em outros modelos treinados, porém para a classe das imagens rotuladas como pneumonia viral o modelo obteve uma sensibilidade de 0,669, devido ao fato que das 148 imagens pertencentes a esta classe 99 foram classificadas corretamente, 32 foram classificadas como pneumonia bacteriana e 17 como imagens de pulmões saudáveis.

Para a classificação binária o modelo treinado obteve uma taxa de acerto para os dados de teste de 90,87%, os erros e acertos cometidos pelo modelo para o caso binário podem ser consultados na matriz de confusão da Tabela 19, assim como as métricas de avaliação do modelo encontradas a partir dos valores de verdadeiro positivo, falso positivo, verdadeiro negativo e falso negativo podem ser consultados na Tabela 19.

Tabela 19 – Matriz de confusão para o problema binário do modelo *Inception V3*.

	Normal	Pneumonia
Normal	213	21
Pneumonia	36	354

Tabela 20 – Métricas de avaliação para o problema de classificação binário do modelo *Inception V3*.

	Sensibilidade	Precisão	F1-Score
Normal	0,910	0,855	0,882
Pneumonia	0,908	0,944	0,925

4.5 ResNet-50

Com a arquitetura de rede neural *ResNet-50* foi-se construído um modelo classificatório com 23.839.379 parâmetros treináveis e 53.120 parâmetros não treináveis e otimizado por 100 épocas utilizando os dados reservados ao treinamento do modelo, onde o gráfico do desenvolvimento do modelo classificatório ao decorrer das épocas pode ser consultado na Figura 25. Ao analisar os gráficos do treinamento é possível perceber que apesar de o modelo ter se ajustado bem aos dados de treinamento, o modelo teve dificuldade em

classificar os dados de validação, obtendo muita variação na acurácia e no valor da função de custo ao decorrer das épocas.

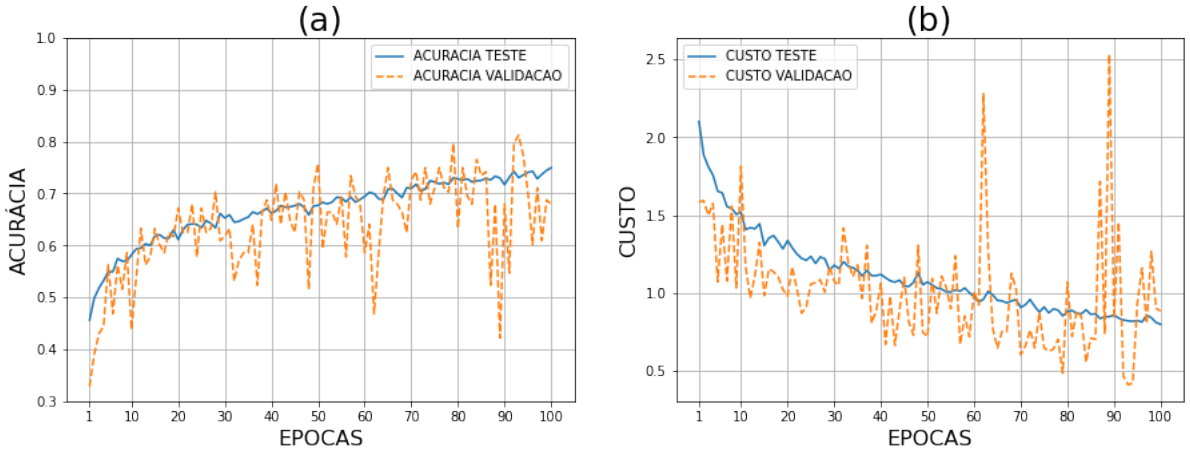

Figura 25 – Gráfico do treinamento modelo *ResNet-50* por 100 épocas para **a)** Acurácia para os dados de treino (linha azul) e de validação (linha laranja) e **(b)** Valor da função de custo para os dados de treino (linha azul) e de validação (linha laranja).

Com o modelo otimizado foi realizado a classificação dos dados reservados ao teste do modelo, como as imagens já eram previamente rotuladas foi possível encontrar a taxa de acerto, sendo esta de 84,14% para os dados de teste. Com as imagens classificadas foi possível construir a matriz de confusão, Tabela 21, esta contendo as classificações realizadas corretamente e erroneamente para classe dos dados de teste.

Tabela 21 – Matriz de confusão para o modelo *ResNet-50*.

	Pneumonia Bacteriana	Normal	Pneumonia Viral
Pneumonia Bacteriana	206	34	2
Normal	0	226	8
Pneumonia Viral	32	23	93

Com a matriz de confusão construída é possível encontrar outras métricas de avaliação do modelo classificatório, além de taxa de acerto. Como o modelo classificatório obteve uma baixa acurácia especifica para a classe pneumonia viral (de 148 imagens 93 foram classificadas corretamente), a taxa de verdadeiros positivos (sensibilidade) será baixa, comparada as outras classes, obtendo também um valor de F_1-*Score* baixo, já que esse é a média harmônica entre a sensibilidade e a precisão.

Tabela 22 – Métricas de avaliação para o modelo *ResNet-50*.

	Sensibilidade	Precisão	F1-Score
Normal	0,966	0,799	0,874
Pneumonia Bacteriana	0,851	0,866	0,858
Pneumonia Viral	0,628	0,903	0,741

Para o problema de classificação de radiografias do tórax de pulmões saudáveis e pulmões com pneumonia a acurácia obtida foi de 85,58% para os dados de teste do modelo. A matriz de confusão construída a partir da classificação dos dados reservados de teste pode ser consultada na Tabela 23.

Tabela 23 – Matriz de confusão para o problema binário do modelo *ResNet-50*.

	Normal	Pneumonia
Normal	226	8
Pneumonia	57	333

Com os valores de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos foi-se possível encontrar as métricas de avaliação do modelo, porém agora para o caso de classificação binária, estes podendo ser consultados na Tabela 24.

Tabela 24 – Métricas de avaliação para o problema de classificação binário do modelo *ResNet-50*.

	Sensibilidade	Precisão	F1-Score
Normal	0,966	0,799	0,874
Pneumonia	0,854	0,977	0,911

4.6 VGG-Inc

O modelo baseado na arquitetura *VGG-16* foi o que obteve o melhor desempenho em classificar as imagens pertencentes ao conjunto de dados reservados ao teste do modelo obtendo uma acurácia de 87,02%, porém apresentou indícios de sobre ajuste dos parâmetros a partir da época 40 no processo de treinamento, sendo assim é proposto um modelo baseado na arquitetura *VGG* e para evitar o sobre ajuste do modelo durante o processo de treinamento é proposto também utilizar os módulos da arquitetura *inception*, aumentando assim consideravelmente a profundidade das camadas do modelo sem o aumento significantemente o custo computacional do treinamento do modelo classificatório.

O modelo proposto utilizando as características dos modelos *VGG* e *Inception* é denominado *VGG-Inc*, o modelo contém uma maior profundidade de camadas com

parâmetros treináveis e um número maior de parâmetros treináveis provenientes de camadas totalmente conectadas. O modelo contem cinco grupos de duas camadas de convolução com tamanho de *kernel* fixos em (3x3) seguidos de uma camada de *Max Pooling*, três módulos *inception* 1 (como ilustrado na Figura 13) e três camadas totalmente conectadas. Os filtros de dimensionalidade utilizados para os grupamentos de convoluções iniciais seguem o padrão da arquitetura *VGG-16*, e os valores utilizados no módulo *inception* podem ser consultados na Tabela 25, o número de neurônios utilizadas nas duas primeiras camadas *dense* foi de 2680 neurônios e a taxa de aprendizagem utilizada para o *Adam* otimizador foi de 1×10^{-5}, estes valores foram encontrados a parir da otimização de hiper-parâmetros do modelo.

Tabela 25 – Parâmetros utilizados para o módulo *inception* durante a construção do modelo.

	1° Módulo Inception	2° Módulo Inception	3° Módulo Inception
1° Grupamento de Convoluções	Tamanho de Kernel: (1x1) Filtros: 64	Tamanho de Kernel: (1x1) Filtros: 128	Tamanho de Kernel: (1x1) Filtros : 192
2° Grupamento de Convoluções	1° Convolução Tamanho de Kernel: (1x1) Filtros: 96 - 2° Convolução Tamanho de Kernel: (3x3) Filtros: 128	1° Convolução Tamanho de Kernel: (1x1) Filtros: 128 - 2° Convolução Tamanho de Kernel: (3x3) Filtros: 192	1° Convolução Tamanho de Kernel: (1x1) Filtros: 95 - 2° Convolução Tamanho de Kernel: (3x3) Filtros: 208
3° Grupamento de Convoluções	1° Convolução Tamanho de Kernel: (1x1) Filtros: 16 - 2° Convolução Tamanho de Kernel: (5x5) Filtros: 32	1° Convolução Tamanho de Kernel: (1x1) Filtros: 32 - 2° Convolução Tamanho de Kernel: (5x5) Filtros: 96	1° Convolução Tamanho de Kernel: (1x1) Filtros: 16 - 2° Convolução Tamanho de Kernel: (5x5) Filtros: 48
4° Grupamento de Convoluções	Max Pooling Janela de Pooling: (3x3) - Convolução Tamanho de Kernel: (1x1) Filtros: 32	Max Pooling Janela de Pooling: (3x3) - Convolução Tamanho de Kernel: (1x1) Filtros: 64	Max Pooling Janela de Pooling: (3x3) - Convolução Tamanho de Kernel: (1x1) Filtros: 64

A arquitetura completa utilizada para o modelo *VGG-Inc* pode ser consultada na Tabela 26 contendo um total de 41 camadas de profundidade (inclusa a camada de entrada), destas 31 contendo parâmetros ajustáveis e um total de 23.081.091 parâmetros que podem ser otimizados durante o processo de treinamento. A definição computacional do módulo *inception* utilizado para a construção do modelo pode ser consultadas no Anexo B.1.

O modelo *VGG-Inc*, descrito na Tabela 26 foi treinada por um total de 100 épocas utilizando os dados reservados ao treinamento do modelo. As acurácias e os valores de função de custo obtidos durante para os dados de treinamento o processo de otimização dos parâmetros treináveis podem ser consultadas na Figura 26.

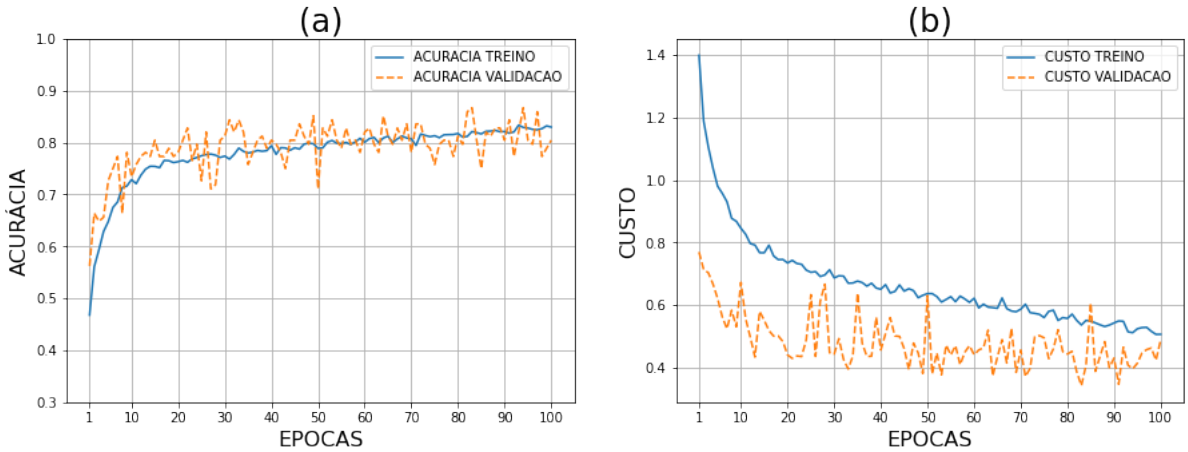

Figura 26 – Gráfico do treinamento do modelo *VGG-Inc* por 100 épocas para **(a)** Acurácia para os dados de treino (linha azul) e de validação (linha laranja) e **(b)** Valor da função de custo para os dados de treino (linha azul) e de validação (linha laranja).

Ao analisar o gráfico do processo de treinamento do modelo *VGG-Inc* é possível notar que os parâmetros treináveis se ajustaram bem aos dados de treinamento, conseguindo também realizar a generalização nas classificações mantendo a taxa de acerto para os dados de treino e validação seguindo a mesma linha de tendência.

Após o processo de treinamento foi possível obter uma taxa de acerto de 91,35% para os dados reservados ao teste do modelo, os erros e acertos do modelo para cada classe podem ser consultados na matriz confusão (Tabela 27).

Tabela 26 – Arquitetura para o modelo *VGG-Inc* proposto a partir das arquiteturas *VGG-16* e *Inception V3*.

Nome da Camada	**Parâmetros**	**Tamanho da Saida**
Camada de Entrada	Tamanho: (224x224x1)	(224x224x1)
1° Convolução	Tamanho de Kernel: (3x3); Filtros: 64; Stride: (1x1)	(224x224x64)
2° Convolução	Tamanho de Kernel: (3x3); Filtros: 64; Stride: (1x1)	(224x224x64)
1° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(112x112x64)
3° Convolução	Tamanho de Kernel: (3x3); Filtros: 128; Stride: (1x1)	(56x56x128)
4° Convolução	Tamanho de Kernel: (3x3); Filtros: 128; Stride: (1x1)	(56x56x128)
2° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(28x28x128)
5° Convolução	Tamanho de Kernel: (3x3); Filtros: 256; Stride: (1x1)	(28x28x256)
6° Convolução	Tamanho de Kernel: (3x3); Filtros: 256; Stride: (1x1)	(28x28x256)
3° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(14x14x256)
7° Convolução	Tamanho de Kernel: (3x3); Filtros: 512; Stride: (1x1)	(14x14x512)
8° Convolução	Tamanho de Kernel: (3x3); Filtros: 512; Stride: (1x1)	(14x14x512)
4° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(7x7x512)
9° Convolução	Tamanho de Kernel: (3x3); Filtros: 512; Stride: (1x1)	(7x7x512)
10° Convolução	Tamanho de Kernel: (3x3); Filtros: 512; Stride: (1x1)	(7x7x512)
5° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(3x3x512)
1° Módulo Inception	Módulo 1	(2x2x256)
2° Módulo Inception	Módulo 1	(2x2x480)
3° Módulo Inception	Módulo 1	(2x2x512)
Flatten	-	2048
1° Dense	Neurônios: 2680; Função de Ativação: ReLU	2680
2° Dense	Neurônios: 2680 ; Função de Ativação: ReLU	2680
3° Dense	Neuônios: 3 ; Função de Ativação: Softmax	3

Tabela 27 – Matriz de confusão para o modelo *VGG-Inc*.

	Pneumonia Bacteriana	Normal	Pneumonia Viral
Pneumonia Bacteriana	228	03	11
Normal	02	216	16
Pneumonia Viral	19	03	126

Com os valores de verdadeiro positivos, verdadeiro negativo, falso positivos e falsos negativos para a classificação multi-classe dos dados reservado ao teste do modelo é possível encontrar outras métricas para a avaliação do modelo classificatório, estas podendo ser consultadas na Tabela 28.

Tabela 28 – Métricas de avaliação para o modelo *VGG-Inc*.

	Sensibilidade	Precisão	F1-Score
Normal	0,923	0,973	0,947
Pneumonia Bacteriana	0,942	0,916	0,929
Pneumonia Viral	0,851	0,824	0,837

Agora, para o problema da classificação binária o modelo classificatório obteve uma taxa de acerto para os dados de teste de 96,15%, onde os erros e os acertos de modelo podem ser consultados na Tabela 29. Com os valores da matriz de confusão é possível obter as métricas de avaliação para os caso de classificação binária (Tabela 30).

Tabela 29 – Matriz de confusão para o problema binário do modelo *VGG-Inc*.

	Normal	Pneumonia
Normal	216	18
Pneumonia	06	384

Tabela 30 – Métricas de avaliação para o problema de classificação binário do modelo *VGG-Inc*.

	Sensibilidade	Precisão	F1-Score
Normal	0,923	0,973	0,947
Pneumonia	0,985	0,955	0,970

É possível notar que o modelo classificatório *VGG-Inc* obteve as melhores métricas de avaliação do modelo, tanto para o caso de classificação multi classe quanto para o caso de classificação binária entre todos os outros modelos treinados, o modelo obteve também

a maior taxa de acerto para os dados de teste esta sendo para o classificação multi classe de 4,33% maior do que a do modelo *VGG-16* e 0,32% maior para a classificação binária comparando a acurácia do *VGG-16*.

Os erros cometidos pelo modelo podem se dar ao fato da qualidade da imagem a ser classificada ser baixa, ou ao se redimensionar a imagem para que fiquem do tamanho da camada de entrada (224x224) algumas imagens podem ter perdido a sua resolução devido ao distorcimento causado na sua redimensionalização.

5 Conclusão

Neste trabalho, foi-se estudado métodos para a classificação automática de imagens provenientes de radiografias do tórax utilizando redes neurais convolucionais, sendo assim, possível classificar imagens em radiografias do tórax que contenham pneumonia viral, bacteriana ou que seja a radiografia de um pulmão sem nenhuma doença, além do problema de classificação multi-classe foi realizado a classificação binária das imagens, onde as imagens de provenientes da radiografia do tórax foram classificadas em pulmões saudáveis ou contendo pneumonia, este podendo ser de origem viral ou bacteriana.

Para chegar ao melhor modelo classificatório possível foi realizada o treinamento e a avaliação de métricas de diferentes redes neurais que de alguma maneira inovaram no campo de estudo das *CNNs*, sendo as arquiteturas escolhidas a *LeNet-5*, *AlexNet*, *VGG-16*, *Inception V3* e *ResNet-50*.

O modelo construído baseado na arquitetura *LeNet-5* obteve uma acurácia de 79,80% para a classificação multi-classe e 86,85% para a classificação binária. O modelo utilizada para a classificação das imagens baseado na arquitetura *AlexNet* obteve uma taxa de acerto de 87,01% para a classificação multi-classe e 92,31% para o problema binário dos dados reservados ao teste do modelo. Já o modelo baseado na arquitetura *VGG-16* foi-se obtido uma acurácia de 87,02% para o problema de classificação multi-classe e 95,83% para a classificação binária. Para o modelo construído com base na arquitetura de rede neural *Inception V3* foi obtido uma taxa de acerto de 84,46% para a classificação multi-classe e 90,87% para a classificação binária. Por fim, o modelo treinado baseado na arquitetura de *CNN ResNet-50* obteve uma taxa de acerto de 84,14% para a classificação multi-classe dos dados de teste e 85,58% para a classificação binária dos dados reservados ao teste do modelo.

A partir dos resultados obtidos de cada modelo classificatório foi proposto uma rede neural convolucional utilizando conceitos das arquiteturas *VGG-16* e *Inception-V3* denominado *VGG-Inc*, esta contendo 41 camadas de profundidade com 31 camadas contendo parâmetros ajustáveis e um total de 23.081.091 parâmetros treináveis. Este modelo classificatório foi otimizado por um total de 100 épocas utilizando os dados reservados ao treinamento do modelo sendo obtido uma taxa de acerto de 91,35%, e o F_1-*Score* obtido foi de 0,947 para a classe de pulmões saudáveis, 0,929 para a classe de pneumonia bacteriana e 0,837 para a classe de pulmões rotulados como contendo pneumonia viral. Para a classificação binária das imagens foi obtido uma acurácia de 96,15%, os F_1-*Scores* obtidos foram de 0,947 para a classe de pulmões saudáveis e 0,970 para a classe de pulmões rotulados com qualquer tipo de pneumonia.

Com as métricas de avaliação obtidas pelo modelo *VGG-Inc* é possível concluir que o modelo classificatório proposto neste trabalho para a classificação de multi-classe de pulmões com pneumonia viral, pneumonia bacteriana e pulmões sem pneumonia e também para a classificação binária, onde o modelo busca apenas classificar uma imagem entre contendo ou não pneumonia, tem potenciais aplicações para a redução do tempo de espera em se obter o resultado do diagnóstico e as possibilidades de ocorrência do erro humano. Porém, se mostrou indispensável a presença do especialista durante o processo de diagnóstico, já que apesar de o modelo ter atingido altas métricas, algumas classificações erradas foram cometidas, essas podendo ser evitadas pelo médico responsável.

Para trabalhos futuros pretende-se iniciar a implementação do modelo classificatório treinado para a classificação de imagens provenientes de novos exames reais, além de coletar as novas imagens para continuar a otimização do modelo.

Referências

[1] R. G. Assunção, W. A. Pereira, and A. G. Abreu, "Pneumonia bacteriana: aspectos epidemiológicos, fisiopatologia e avanços no diagnóstico," *Rev Inv Biomédica*, vol. 10, no. 1, pp. 83–91, 2018.

[2] WHO, "Pneumonia." https://www.who.int/health-topics/pneumonia#tab=tab_1, 2019. Acessado em 08/22/2021.

[3] "Pneumonia, Biblioteca Virtual em Saúde MS." https://bvsms.saude.gov.br/pneumonia-5/, 2011. Acessado em 08/22/2021.

[4] R. Takahashi and Y. Kajikawa, "Computer-aided diagnosis: A survey with bibliometric analysis," *International Journal of Medical Informatics*, 2017.

[5] Y. Zhou, H. Chen, Y. Li, Q. Liu, X. Xu, S. Wang, P.-T. Yap, and D. Shen, "Multi-task learning for segmentation and classification of tumors in 3D automated breast ultrasound images," *Medical Image Analysis*, 2021.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] D. Kermany, K. Zhang, and M. Goldbaum, "Large dataset of labeled optical coherence tomography (oct) and chest x-ray images," 2018.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[13] O. Ruuskanen, E. Lahti, L. C. Jennings, and D. R. Murdoch, "Viral pneumonia," *The Lancet*, vol. 377, no. 9773, pp. 1264–1275, 2011.

[14] M. T. K. Kotsubo, E. Marchiori, and A. C. P. d. Azevedo, "Estudo dosimétrico de radiografias de tórax com o emprego de técnicas de alta quilovoltagem," *Radiologia Brasileira*, 2003.

[15] L. d. S. L. Lauand, E. B. de Souza Junior, B. J. Andrade, and S. R. S. Sprovieri, "Contribuição da interpretação da radiografia simples de tórax na sala de emergência," *Arquivos Médicos dos Hospitais e da Faculdade de Ciências Médicas da Santa Casa de São Paulo*, 2008.

[16] G. Lacey, L. Berman, and S. Morley, *Radiografia Do Tórax Um Guia Prático*. Elsevier Brasil, 2011.

[17] J. McCarthy, "What is Artificial Intelligence?," 2007.

[18] X. Yao and Y. Liu, *Machine Learning*. 2014.

[19] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018.

[20] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," 2015.

[21] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, p. 31–44, Mar. 1996.

[22] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[23] Aggarwal, Charu C, *Neural Networks and Deep Learning*, vol. 10. Springer, 2018.

[24] Y. Wang, Y. Li, Y. Song, and X. Rong, "The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition," *Applied Sciences*, 2020.

[25] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, PMLR, 2011.

[27] C. M. Bishop, *Pattern Recognition*, vol. 128. Springer, 2006.

[28] S. E. Umbaugh, *Computer Imaging: Digital Image Analysis and Processing*. CRC Press, 2005.

[29] X. Zhang, Y. Wang, N. Zhang, D. Xu, and B. Chen, "Research on scene classification method of high-resolution remote sensing images based on rfpnet," *Applied Sciences*, vol. 9, no. 10, 2019.

[30] E. W. Weisstein, "Convolution." https://mathworld.wolfram.com/Convolution.html. Acessado em 09/25/2021.

[31] E. R. C. Huacarpuma, M. Miazaki, and V. Gryzak, "Segmentação de imagens de tomografia computadorizada de estruturas pélvicas utilizando técnicas de deep learning,"

[32] A. Ng, K. Katanforoosh, and Y. Mourri, "Convolutional Neural Networks, Aula 1: Convolutions Over Volume, [DeepLearning.ai]." https://www.coursera.org/learn/convolutional-neural-networks. (Acesso em 28/11/2021).

[33] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed Pooling for Convolutional Neural Networks," in *Rough Sets and Knowledge Technology*, Springer International Publishing, 2014.

[34] Stanford Vision Lab, "Imagenet Large Scale Visual Recognition Challenge (ILSVRC)." https://www.image-net.org/challenges/LSVRC/index.php, 2020. (Acesso em 17/11/2021).

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[37] A. Ng, K. Katanforoosh, and Y. Mourri, "Convolutional Neural Networks, Aula 2: Inception Network Motivation, [DeepLearning.ai]." https://www.coursera.org/learn/convolutional-neural-networks. (Acesso em 21/11/2021).

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[39] C. L. d. Castro and A. P. Braga, "Aprendizado supervisionado com conjuntos de dados desbalanceados," *Sba: Controle & Automação Sociedade Brasileira de Automatica*, vol. 22, pp. 441–466, 2011.

[40] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, 2020.

[41] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *European conference on information retrieval*, Springer, 2005.

[42] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis," in *ICDAR*, vol. 3, 2003.

[43] M. Claro, L. Vogado, J. Santos, and R. Veras, "Utilização de técnicas de data augmentation em imagens: Teoria e prática," pp. 47–71, SBC, Sept. 2020.

[44] Keras API, "ImageDataGenerator." https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator, 2021. (Acesso em 11/11/2021).

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014.

[46] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," 2018.

Apêndices

APÊNDICE A – Arquiteturas das redes neurais convolucionais

A.1 *LeNet-5*

Tabela 31 – Arquitetura da *LeNet 5*, todas as camadas de convolução tem como função de ativação a função *tanH*; Adaptado de *LeCun et al.* (1998).

Nome da Camada	**Parâmetros**	**Tamanho da Saida**
Camada de Entrada	Tamanho: (32x32x1)	(32x32x1)
1° Convolução	Tamanho de Kernel: (5x5); Filtros: 6; Stride: (1x1)	(28x28x6)
1° Pooling	Tipo de Pooling: Average Pooling; Janela de Pooling: (2x2); Stride: (2x2)	(14x14x6)
2° Convolução	Tamanho de Kernel: (5x5); Filtros: 16; Stride: (1x1)	(10x10x16)
2° Pooling	Tipo de Pooling: Average Pooling; Janela de Pooling: (2x2); Stride: (2x2)	(5x5x16)
3° Convolução	Tamanho de Kernel: (5x5); Filtros: 120; Stride: (1x1)	(1x1x120)
1° Dense	Número de Neuronios: 84; Função de Ativação: tanH	84
2° Dense	Número de Neuronios: 10; Função de Ativação: Softmax	10

A.2 *AlexNet*

Tabela 32 – Arquitetura da rede neural convolucional *AlexNet*, todas as camadas de convolução tem como função de ativação a função *ReLU*; Adaptado de *Krizhevsky, Sutskever & Hilton* (2014).

Nome da Camada	**Parâmetros**	**Tamanho da Saida**
Camada de Entrada	Tamanho: (224x224x3)	(224x224x3)
1° Convolução	Tamanho de Kernel: (11x11); Filtros: 96; Stride: (4x4)	(55x55x96)
1° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(27x27x96)
2° Convolução	Tamanho de Kernel: (5x5); Filtros: 256; Stride: (1x1)	(27x27x256)
2° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(13x13x256)
3° Convolução	Tamanho de Kernel: (3x3); Filtros: 384; Stride: (1x1)	(13x13x384)
4° Convolução	Tamanho de Kernel: (3x3); Filtros: 384; Stride: (1x1)	(13x13x384)
5° Convolução	Tamanho de Kernel: (3x3); Filtros: 256; Stride: (1x1)	(13x13x256)
4° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(6x6x256)
Flatten	-	9216
1° Dense	Neurônios: 4096; Função de Ativação: ReLU	4096
2° Dense	Neurônios: 4096; Função de Ativação: ReLU	4096
3° Dense	Neurônios: 1000; Função de Ativação: Softmax	1000

A.3 *VGG-16*

Tabela 33 – Arquitetura da *VGG-16*, todas as camadas de convolução tem como função de ativação a função *ReLU*; Adaptado de *Simonyan & Zisserman* (2016).

Nome da Camada ou Bloco	**Parâmetros**	**Tamanho da Saída**
Camada de Entrada	Tamanho: (224x224x3)	(224x224x3)
2 x 1° Grupo de Convolução	Tamanho de Kernel: (3x3); Filtros: 64; Stride: (1x1)	(224x224x64)
1° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(112x112x64)
2 x 2° Grupo de Convolução	Tamanho de Kernel: (3x3); Filtros: 128; Stride: (1x1)	(112x112x128)
2° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(56x56x128)
2 x 3° Grupo de Convolução	Tamanho de Kernel: (3x3); Filtros: 256; Stride: (1x1)	(56x56x256)
3° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(28x28x256)
3x 4° Grupo de Convolução	Tamanho de Kernel: (3x3); Filtros: 512; Stride: (1x1)	(28x28x512)
4° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(14x14x512)
3 x 5° Grupo de Convolução	Tamanho de Kernel: (3x3); Filtros: 512; Stride: (1x1)	(14x14x512)
5° Pooling	Tipo de Pooling: Max Pooling; Janela de Pooling: (3x3); Stride: (2x2)	(7x7x512)
Flatten	-	25088
1° Dense	Neurônios: 4096; Função de Ativação: ReLU	4096
2° Dense	Neurônios: 4096; Função de Ativação: ReLU	4096
3° Dense	Neurônios: 1000; Função de Ativação: Softmax	1000

A.4 *Inception V3*

Tabela 34 – Arquitetura da *Inception-V3*, todas as camadas de convolução tem como função de ativação a função $ReLU$; o Módulo 1 é retratado na Figura 13, o Módulo 2 é retratado na Figura 14, o Módulo 3 é retratado na Figura 15; Adaptado de *Szegedy et al.* (2016).

Nome da Camada ou Módulo	**Parâmetros**	**Tamanho da Saída**
Camada de Entrada	Tamanho: (299x299x3)	(299x299x3)
1° Convolução	Tamanho de Kernel: (3x3); Filtros: 32; Stride: (2x2)	(149x149x32)
2° Convolução	Tamanho de Kernel: (3x3); Filtros: 32; Stride: (1x1)	(147x147x32)
3° Convolução	Tamanho de Kernel: (3x3); Filtros: 64; Stride: (1x1)	(147x147x64)
1° Pooling	Tipo: Max Pooling; Janela de Pooling: 3x3; Stride: (2x2)	(73x73x64)
4° Convolução	Tamanho de Kernel: (3x3); Filtros: 80; Stride: (1x1)	(71x71x80)
5° Convolução	Tamanho de Kernel: (3x3); Filtros: 192; Stride: (2x2)	(35x35x192)
6° Convolução	Tamanho de Kernel: (3x3); Filtros: 288; Stride: (1x1)	(35x35x288)
3 x Módulo Inception	Módulo 1	(17x17x768)
5 x Módulo Inception	Módulo 2	(8x8x1280)
2 x Módulo Inception	Módulo 3	(8x8x2048)
2° Pooling	Tipo: Average Pooling; Janela de Pooling: 8x8; Stride: (2x2)	(1x1x2048)
Flatten	-	2048
Dense	Neurônios: 1000; Função de Ativação: Softmax	1000

A.5 *ResNet-50*

Tabela 35 – Arquitetura da *ResNet-50*, todas as camadas de *pooling* e convolução tem *stride* de $(2x2)$, todas as camadas de convolução tem como função de ativação a função $ReLU$; Adaptado de *He et al.* (2016).

Nome da Camada ou Bloco	**Parâmetros**
Camada de Entrada	Tamanho: (224x224x3)
1° Convolução	Tamanho do Kernel: (7x7); Filtros: 64
1° *Pooling*	Tipo: Max Pooling; Janela de Pooling: (3x3)
3 x 1º Bloco de Convolução	Convolução 1: Tamanho do Kernel: (1x1); Filtros: 64; Convolução 2: Tamanho do Kernel: (3x3); Filtros: 64; Convolução 3: Tamanho do Kernel: (1x1); Filtros: 256;
4 x 2º Bloco de Convolução	Convolução 1: Tamanho do Kernel: (1x1); Filtros: 128; Convolução 2: Tamanho do Kernel: (3x3); Filtros: 128; Convolução 3: Tamanho do Kernel: (1x1); Filtros: 512;
6 x 3º Bloco de Convolução	Convolução 1: Tamanho do Kernel: (1x1); Filtros: 256; Convolução 2: Tamanho do Kernel: (3x3); Filtros: 256; Convolução 3: Tamanho do Kernel: (1x1); Filtros: 1024;
3 x 4º Bloco de Convolução	Convolução 1: Tamanho do Kernel: (1x1); Filtros: 512. Convolução 2: Tamanho do Kernel: (3x3); Filtros: 512. Convolução 3: Tamanho do Kernel: (1x1); Filtros: 2048.
2° Pooling	Tipo: Average Pooling; Janela de Pooling: (2x2);
Flatten	-
Dense	Neurônios: 1000; Função de Ativação: Softmax

APÊNDICE B – Código *VGG-Inc*

B.1 Definição do módulo *inception*

```
def inceptionModule(x, filters1x1, filters3x3r, filters3x3, filters5x5r,
    filters5x5, filtersPoll, name=None):
    conv1x1 = Convolution2D(filters = filters1x1,
                            kernel_size = (1,1),
                            padding='same',
                            ativation='relu')(x)

    conv3x3 = Convolution2D(filters = filters3x3r,
                            kernel_size = (1, 1),
                            padding='same',
                            activation='relu')(x)

    conv3x3 = Convolution2D(filters = filters3x3,
                            kernel_size = (3, 3),
                            padding='same',
                            activation='relu')(conv3x3)

    conv5x5 = Convolution2D(filters = filters5x5r,
                            kernel_size = (1, 1),
                            padding='same',
                            activation='relu')(x)

    conv5x5 = Convolution2D(filters = filters5x5,
                            kernel_size = (5, 5),
                            padding='same',
                            activation='relu')(conv5x5)

    poolProj = MaxPooling2D((3, 3),
                            strides=(1, 1),
                            padding='same')(x)

    poolProj = Convolution2D(filtersPoll,
                            filters = (1, 1),
                            padding='same',
                            activation='relu')(poolProj)

    output = concatenate([conv1x1, conv3x3, conv5x5, poolProj],
                         axis=3,name=name)

    return output
```

B.2 Definição do modelo *VGG-Inc*

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras.layers import Input, Convolution2D, MaxPooling2D, concatenate
from keras.layers import Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam

input_layer = Input(shape=(224, 224, 1))
# 1 grupo
x = Convolution2D(filters = 64, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(input_layer)
x = Convolution2D(filters = 64, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

# 2 grupo
x = Convolution2D(filters = 128, kernel_size =  (3,3),
                  strides=(1, 1), activation='relu')(x)
x = Convolution2D(filters = 128, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

# 3 grupo
x = Convolution2D(filters = 256, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = Convolution2D(filters = 256, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

# 4 grupo
x = Convolution2D(filters = 512, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = Convolution2D(filters = 512, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

# 5 grupo
x = Convolution2D(filters = 512, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = Convolution2D(filters = 512, kernel_size = (3,3),
                  strides=(1, 1), activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

# 1 modulo inception
x = inceptionModule(x,
```

```
                    filters1x1=64,
                    filters3x3r=96,
                    filters3x3=128,
                    filters5x5r=16,
                    filters5x5=32,
                    filtersPoll=32,
                    name='inception1')

# 2 modulo inception
x = inceptionModule(x,
                    filters1x1=128,
                    filters3x3r=128,
                    filters3x3=192,
                    filters5x5r=32,
                    filters5x5=96,
                    filtersPoll=64,
                    name='inception2')

# 3 modulo inception
x = inceptionModule(x,
                    filters1x1=192,
                    filters3x3r=95,
                    filters3x3=208,
                    filters5x5r=16,
                    filters5x5=48,
                    filtersPoll=64,
                    name='inception3')

x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(units = 2680, activation = 'relu')(x)
x = Dropout(0.2)(x)
x = Dense(units = 2680, activation = 'relu')(x)
x = Dense(3, activation = 'softmax', name='output')(x)

model  = Model(input_layer, x)
model.compile(loss = 'categorical_crossentropy',
              optimizer = Adam(learning_rate = 1e-05),
              metrics = ['accuracy'])
```