UNIVERSIDADE METODISTA DE PIRACICABA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DO MOVIMENTO HUMANO

MOVIMENTO HUMANO E INTELIGÊNCIA ARTIFICIAL: DESVIOS ESCOLIÓTICOS NA COLUNA LOMBAR

José Ricardo Lourenço de Oliveira

2022

TESE DE DOUTORADO

José Ricardo Lourenço de Oliveira

MOVIMENTO HUMANO E INTELIGÊNCIA ARTIFICIAL: DESVIOS ESCOLIÓTICOS NA COLUNA LOMBAR

Tese apresentada junto ao Programa de Pós-Graduação em Ciências do Movimento Humano, da Universidade Metodista de Piracicaba, para obtenção do Título de Doutor em Ciências do Movimento Humano.

Orientador: Prof. Dr. Guanis de Barros Vilela Junior

Piracicaba

2022

Ficha Catalográfica elaborada pelo Sistema de Bibliotecas da UNIMEP
Bibliotecária: Joyce Rodrigues de Freitas - CRB-8/10115.

O48m

Oliveira, José Ricardo Lourenço de
Movimento humano e inteligência artificial: desvios escolióticos na coluna lombar /José Ricardo Lourenço de Oliveira- 2022.
144f. ; 30 cm.

Orientador: Prof. Dr. Guanis de Barros Vilela Junior.
Tese (Doutorado) – Universidade Metodista de Piracicaba, Ciências do Movimento Humano, Piracicaba, 2022.

1. Inteligência Artificia. 2. Saúde. 3.Escoliose. I. Oliveira, José Ricardo Lourenço de. II. Título.

CDD – 153

DEDICATÓRIA

Este momento ímpar da minha vida retrata a presença e a glória de Deus sobre tudo e todos, portanto é a principal dedicação deste estudo.

Todavia, algumas brilhantes pessoas, pontuaram minha trajetória de vida e me impulsionaram minha caminhada, são elas:

Minha mãe - Sra. Maria Aparecida Lourenço (in memoriam), pelo provimento e amor incondicional.

Minha Avó - Sra. Jandyra Corrêa Gomes (in memoriam), por acreditar em mim e apostar que um dia eu poderia ser “doutor".

Minha esposa - Profa. Dra. Heleise Faria dos Reis de Oliveira - pela força, fé e incansável propósito de superarmos todos os obstáculos que nos ocorria nestes 27 anos juntos.

Minha irmã - Juliana Lourenço Evangelista Oses, por demonstrar seu carinho sem igual e me apoiar sempre.

E por último a fusão de todas essas mulheres fortes e maravilhosas, a minha filha Isabella Faria dos Reis de Oliveira, o meu orgulho de vida, a qual é minha razão de ser e existir.

AGRADECIMENTOS

Em primeiro lugar quero agradecer ao órgão fomentador da minha pesquisa, a CAPES, a Coordenação de Pós-graduação de Ciências do Movimento Humano - CMH da Universidade Metodista de Piracicaba - UNIMEP, especialmente a Profa. Dra. Rozangela Verlengia pelo sempre pronto suporte, bem como, cada professor do curso, os quais acrescentaram um pouco mais ao meu novo saber. Agradecer também, meus colegas de curso, com foco naqueles pertencentes ao nosso núcleo de pesquisa pela sempre presente conversa, embora tivéssemos a distância - e foram muitas.

Aos meus colegas de trabalho que tanto sentiram minha falta durante esse processo de doutoramento, no qual tive de me afastar, agradeço particularmente a fiel amiga Nádia Leão pelo seu compromisso e dedicação.

Em particular agradeço meu amigo de todas as horas o incansável “proseador” que estimo e considero parte da minha família, pois nossa história se cruzou há 24 anos e nesse meio tempo só se solidificou. Este amigo, transcende a amizade e verdadeiramente transforma-se em família de fato, pois daqui para frente passo ser descendente dele e de sua sabedoria, ou “ao menos parte dela” (…) com muito orgulho e satisfação. Sem dúvidas o Prof. Dr. Guanis de Barros Vilela Junior é essa pessoa, amorosa, zelosa e cuidadosa para com os seus, por tudo isso tudo e mais um pouco, deixo aqui meu muito obrigado de coração.

E por fim agradeço a toda minha família, tia Teresinha, sobrinhos, cunhadas e cunhados, com distinção a minha sempre apoiadora a quem é um exemplo de história de vida, minha sogra, a Profa. Dra. Marília Faria de Miranda.

RESUMO

Introdução: As Ciências do Movimento Humano tem avançado ao longo dos anos no cenário da inovação e tecnologia, diante disso, buscou-se a Inteligência Artificial (IA) como instrumento a ser empregue para essa inovação contínua. Sabe-se que a IA têm dominado o século XXI e sua grandeza tem revolucionado o âmbito computacional e vivencial do Ser Humano, aperfeiçoando a aprendizagem das máquinas de forma célere, devido a uma maior rapidez de aprendizado nas tomadas de decisões iniciada já na programação. O uso da IA tem multiplicado o conhecimento na ciência, indústria, comércio, serviços e principalmente na área da saúde, a qual tem beneficiado suas intervenções de forma precisa e eficaz. Desta forma, pensou-se na seguinte questão problema: é possível desenvolver e validar uma rede neural convolucional - CNN (*Convolutional Neural Networks*) capaz de identificar com eficiência, desvios escolióticos patológicos na coluna lombar? **Objetivo:** criar, desenvolver e validar um algoritmo inteligente, destinado à avaliação de desvios posturais da coluna lombar, por meio de uma CNN. **Métodos:** pesquisa descritiva de caráter pré-diagnóstico e métodos de IA aplicados, utilizando códigos em linguagem Python®, bem como, bibliotecas suplementares para a geração da CNN que foi arquitetada com 3 cenários. No primeiro cenário a codificação de programação partiu da modelo referência, sofrendo incrementos para a extração de novas imagens de Raio X (RX). No segundo cenário, pôde-se aprofundar os números de conexões inter e intra camadas da CNN e por fim o terceiro cenário, refere-se à incrementação de duas camadas de transposição de matrizes com a finalidade de melhorar a performance de identificação de existência de desvio patológicos da coluna lombar. Inicialmente obteve-se 2897 imagens cedidas e de banco de dados integralizando ao final 579400 diferentes imagens. As etapas para a estruturação da CNN, compreendem além da consolidação da base de dados, as fases de treinamento e teste do algoritmo**. Resultados:** Os diferentes cenários geraram, inicialmente, cerca de 260 milhões de parâmetros e com a otimização da CNN, o cenário 3 atingiu melhor performance 1,8 milhões de parâmetros. As métricas finais da CNN, apresentaram acurácia de 96%; precisão de 98%; sensibilidade de 91%; a especificidade 99%. Adicionalmente foram obtidos os índices FMI=0,934; MCC=0,894 e IY=0,877, que garantem maior robustez na eficiência da CNN na identificação de desvios patológicos de coluna lombar. **Considerações Finais:** A CNN desenvolvida e validada corroborou com a hipótese proposta mostrando elevada confiabilidade no diagnóstico de desvios lombares escolióticos. O presente algoritmo provavelmente apresenta um grande potencial de aplicação nas práticas circunscritas às ciências do movimento humano e podendo contribuir valorosamente em clínicas, estúdios e academias.

Palavras-chave: Inteligência Artificial; Saúde; Escoliose.

ABSTRACT

Introduction: The Human Movement Sciences have advanced over the years in the innovation and technology scenario, in view of this, Artificial Intelligence (AI) was sought as a tool to be used for this continuous innovation. It is known that AI has dominated the 21st century and its greatness has revolutionized the computational and experiential scope of the Human Being, improving machine learning quickly, due to a greater speed of learning in decision-making already initiated in programming. The use of AI has multiplied knowledge in science, industry, commerce, services and especially in the health area, which has benefited its interventions in a precise and effective way. In this way, the following problem question was thought: is it possible to develop and validate a convolutional neural network - CNN (Convolutional Neural Networks) capable of efficiently identifying pathological scoliotic deviations in the lumbar spine? **Objective**: to create, develop and validate an intelligent algorithm, for the evaluation of postural deviations of the lumbar spine, through a CNN. **Methods**: pre-diagnosis descriptive research and applied AI methods, using codes in the Python® language, as well as supplementary libraries for generating the CNN, which was designed with 3 scenarios. In the first scenario, the programming encoding started from the reference model, suffering increments for the extraction of new X-Ray (RX) images. In the second scenario, it was possible to deepen the numbers of connections inter and intra layers of CNN and finally the third scenario, refers to the increase of two layers of transposition of matrices with the purpose of improving the performance of identification of the existence of deviation pathologies of the lumbar spine. Initially, 2897 images provided and from the database were obtained, finally adding up to 579400 different images. The steps for structuring the CNN include, in addition to consolidating the database, the training and testing phases of the algorithm. **Results**: The different scenarios initially generated about 260 million parameters and with the CNN optimization, scenario 3 achieved a better performance of 1.8 million parameters. CNN's final metrics showed an accuracy of 96%; 98% accuracy; 91% sensitivity; the specificity 99%. Additionally, the indices FMI=0.934; MCC=0.894 and IY=0.877, which ensure greater robustness in the efficiency of CNN in identifying pathological deviations of the lumbar spine. Final Considerations: The developed and validated CNN corroborated the proposed hypothesis, showing high reliability in the diagnosis of scoliotic lumbar deviations. The present algorithm probably has a great potential for application in practices limited to the sciences of human movement and can make a valuable contribution in clinics, studios, and academies.

Keywords: Artificial Intelligence; Health; Scoliosis.

LISTA DE ABREVIATURAS E SIGLAS

CID	**Código Internacional de Doenças**
CNN	**Redes Neurais Convolucionais (Convolutional Neural Networks)**
CMH	**Ciências do Movimento Humano**
EIA	**Escoliose Idiopática Adolescente**
IA	**Inteligência Artificial**
IoT	**Internet das Coisas (Internet of Things)**
F	**F-Score**
RNA	**Rede Neural Artificial**
RX	**Raio X**
IDE	**Ambiente de Desenvolvimento Integrado**
GAN	**Redes Generativas Adversárias (Generative Adversarial Networks)**
WHO	**World Health Organization**

LISTA DE ILUSTRAÇÕES

Figuras

GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

Na atualidade vê-se grande movimentação global no sentido de automatizar processos para um futuro digital que melhore a qualidade de vida das pessoas, tais como, adultos, idosos que continuam laborando, alunos e pacientes clínicos, especialmente aqueles inseridos no mundo do trabalho.

Neste sentido, emerge a necessidade de se discutir acerca dos avanços tecnológicos na área da saúde, tais como; a telemedicina, inteligência artificial, rede neurais artificiais, internet das coisas (IoT), *software* de gestão para clínicas, prontuário eletrônico, diagnósticos personalizados, robôs cirúrgicos e nanomedicina, tudo isso, tudo isso está a impulsionando a tecnologia digital na evolução dos cuidados relativos de saúde.

Sendo assim, o foco deste estudo parte da estimativa de McAviney, *et al.* (2020) que até 2050, a proporção da população mundial com idade acima dos 60 anos quase dobrará, o que colocará uma pressão significativa nos sistemas de saúde em todo o mundo, em virtude da fragilidade física, advinda com o avanço da idade. Neste sentido, destaca-se particularmente a dor lombar de acordo com o CID M54.5 (WHO, 2022), classificada como dor lombar baixa, considerada como uma das quatro doenças mais comuns que se tornaram um problema econômico na área da saúde em muitos países atingindo pessoas adultas e inclusive idosos entre grupos populacionais (JACHSTET *et al*, 2022).

A despeito disso, uma condição alarmante que chama a atenção para estudiosos da saúde, pois grande parte da humanidade sofre com problemas de coluna, como evidenciado por diversos autores tais como: Wu *et al*. (2017) Chou, *et al*. (2016) e França, *et al*. (2010), pois estima-se que entre 70% a 80% dos

adultos, terão de um a dois episódios de algias na região lombar durante suas vidas.

Com vistas a todo este entendimento, instrumentalizou-se meios, os quais, pudessem aprimorar o diagnóstico de imagens de raio x, a ponto de alimentar uma Rede Neural Artificial (RNA) que são modelos computacionais, inspirados na rede neural humana, treinando-a para fornecer dados concretos e grande assertividade aos desvios da coluna lombar, fato este que pode ser corroborado por Zhou *et al*. (2019) que traz aplicações distintas da utilização de RNA, destacando as similaridades entres as vértebras da coluna vertebral e neste caso, a tendo a região lombar também, como referência.

Outros pesquisadores, de diversas áreas, têm endossado fortemente um momento muito importante para desdobramentos em ciências aplicadas no que se diz respeito a criação de novas RNA interligadas a saúde, entre eles revela-se o "novo método automatizado de segmentação vertebral que permite a detecção precisa do alinhamento sagital, sem restrições impostas pela qualidade da imagem ou tipo de patologia" (ZHANG *et al*., 2020, p. 387).

Nesta seara, Yasaka *et al*. (2020) ao destacar a previsão da densidade mineral óssea da tomografia computadorizada: aplicação de aprendizado profundo com uma rede neural, como também, Löffler *et al*. (2020) onde menciona sobre a triagem da osteoporose e a baixa densidade óssea em pacientes com afrouxamento de parafusos, após instrumentação semirrígida na coluna lombar.

Já no campo intervencionista surge Kausch *et al*. (2020), com posicionamento automático de projeções com padrão em cirurgia ortopédica, Fan *et al*. (2020), que trata da reconstrução lombossacral baseada em aprendizado profundo, para a previsão de dificuldade de discectomia endoscópica transforaminal percutânea, ainda Biswas *et al*. (2020) com o projeto de Implante

Espinhal personalizado para cada paciente ou também, Casagrande e Vilela Júnior (2015) que discute a Determinação da posição tridimensional da cúpula acetabular em pacientes com artroplastia total do quadril.

Neste cenário, encontra-se grande viabilidade de desenvolvimento de uma RNA, utilizando imagens de RX, ferramenta viável, popular e democrática, pois possui baixo custo e amplamente disponível em todo o mundo, particularmente, propõe-se a análise de desvios posturais que são considerados patológicos da coluna vertebral, sendo potencialmente incapacitantes para a prática de atividade física do cotidiano e exercícios físicos regulares.

Para tanto, propõe-se a Rede Neural Convolucional (Convolutional Neural Network - CNN), chamada de mecanismo de convolução, em que cada uma das imagens da base de dados é mapeada em busca de suas características morfológicas e de canais de cor / tons cinza, na sequência, filtros são aplicados, rastreando toda a extensão da imagem em busca de valores de diferenciação de pixels, gerando a partir dos mesmos, mapas que são novamente processados, até que se possa ter um mapa final que possibilite a classificação do desvio e das diferenças entre diferentes imagens.

A ferramenta de CNN foi extensamente utilizada nos últimos dois anos em decorrência da necessidade de respostas rápidas e asseguradas em detrimento da pandemia da Covid-19, como pode ser observado em Sousa *et al* (2022), Hossain *et al*. (2022), Laddha e Kumar (2022), Yaşar e Ceylan (2022) e Tiwari, Singhal, e Dhankhar (2022) entre outros; tal fato colaborou com o aumento da utilização de algoritmos inteligentes, contribuindo para o objetivo desta pesquisa, que é criar, desenvolver e otimizar um algoritmo inteligente, destinado à avaliação de desvios posturais da coluna lombar, por meio de uma CNN.

Desta forma, pensou-se na seguinte questão problema: é possível desenvolver uma CNN, capaz de identificar e classificar com eficiência, desvios escolióticos patológicos na coluna lombar?

1.1 Justificativa

Um dos grandes desafios da área da saúde, mormente das Ciências do Movimento Humano (CMH) é melhorar a eficiência das avaliações e diagnósticos relativos à postura humana, de acordo com Deutsch, Gill-Body e Schenkman (2022) o papel e a conciliação da ciência do movimento com a linguagem da Classificação Internacional de Funcionalidade (CIF) é evidenciado na importância de cada perspectiva com o cuidado às individualidades.

Nesta perspectiva, buscou-se possibilidades, as quais pudessem nortear pesquisas de cunho científico inovador e de vanguarda tecnológica, utilizando a CNN como alicerce desse constructo.

Conquanto este estudo torna-se relevante, pois a grande área do movimento humano carece de meios auxiliares de análise, identificação, diagnósticos e avaliações de saúde, sobretudo nas etapas interpretativas de imagens de raio x, pois oferece maior agilidade com eficiência, acurácia, precisão entre outras métricas de avaliação.

Diariamente pessoas de toda sorte, sejam alunos e/ou pacientes chegam aos estúdios, academias e centro de fisioterapia com imagens de raio x sem laudos e solicitando parecer a respeito de qual movimento é melhor ser conduzido naquele caso.

Sendo assim, resolveu-se desenvolver, implementar e validar uma rede

neural baseada em Inteligência Artificial (IA), a fim de favorecer a prática do profissional de saúde.

1.2 Objetivos

1.2.1 Objetivo Geral:

- Criar, desenvolver e validar um algoritmo inteligente, destinado à avaliação de desvios posturais da coluna lombar, por meio de uma *Convolutional Neural Network* (CNN).

1.2.2 Objetivos Específicos:

- Desenvolver e validar a CNN para identificação de desvios escolióticos patológicos da coluna lombar.
- Aferir a eficiência da CNN desenvolvida, por meio da acurácia, precisão, sensibilidade, especificidade, *Fowlkes-Mallows* (FMI), Coeficiente de Correlação de *Matthews* (MCC) e Índice de *Youden* (IY).

1.3 Hipótese

O Algoritmo com CNN avaliará com eficiência os desvios escolióticos de coluna lombar, caracterizando-os como normais ou patológicos.

2 FUNDAMENTAÇÃO: O MOVIMENTO HUMANO E SUAS INTERAÇÕES

Ao longo da história grandes realizações na saúde, moveram a humanidade para a própria evolução, tendo como exemplo a criação da vacina em 1796, da anestesia em 1842, do Raio X em 1895, da cultura de tecidos em 1906, da compreensão acerca do colesterol em 1912, do surgimento de antibióticos em 1929, ou ainda dos primeiros passos para o deciframento do DNA humano em 1953, segundo (FRIEDMAN; FRIEDLAND 2000).

Neste sentido, cabe apontar que a natureza epistemológica das CMH, tem sido uma das grandes descobertas da atualidade no campo da saúde, principalmente, enquanto o rol de possibilidades e abrangências inerentes a suas práticas transfenomenais de acordo com (VILELA JUNIOR, 2015).

O movimento humano está para a saúde, tal como para sua própria essência, destacando-se como cuidados efetivos, o bem-estar, a promoção e reabilitação de condições físicas e emocionais, buscando a tão almejada qualidade de vida.

Neste estudo o foco foi direcionado para uma das tantas viabilidades da CMH que é o caso dos desvios posturais responsável por lombalgias, evidenciado por Wasinpongwanich, Nopsopon e Pongpirul (2021), Taguchi, Nakano e Nozawa (2021) e Noh (2021), os quais apontam a necessidade de aprofundamentos de estudos nesta área.

Entre os desvios posturais o que recebeu destaque, foi a escoliose, esta por sua vez pode ser herdada de forma congênita ou idiopática e ainda adquirida a partir dos hábitos posturais, distúrbios neuromusculares ou até degenerativa no caso de adultos com o avanço a idade, entre outros padrões influenciados por

fatores ambientais, situação aludida por Negrini *et al*. (2022), Lenz *et al*. (2021) e Bradko *et al*. (2021).

A escoliose é definida como curvatura da coluna vertebral no plano coronal. É tipicamente acompanhada por um grau variável de rotação da coluna vertebral. Sendo uma curvatura anormal para um lado ou outro da coluna.

A escoliose enquanto patologia deve ser constatada e tratada a ponto de promover o bem-estar, por meio da reabilitação e práticas de exercícios físicos direcionados, respeitando as características de cada indivíduo, conforme Leonard (2022).

Um ponto de atenção a observar, segundo Marinho e Da Pas (2022), grande parte de adolescentes, possuem escoliose idiopática adolescente - EIA, afetando até 5% dessa população; neste contexto é provável que parte destes adolescentes ao atingirem a fase adulta continuem apresentando escoliose lombar caso a mesma não tenha sido tratada anteriormente.

Por isso, observou-se unicamente a coluna vertebral e particularmente a região lombar, pois a curvatura lombar é evidenciada como patológica em imagens de RX, direcionamento dado por outros pesquisadores, tais como Wong, Cheung e Cheung, (2022), Bizzoca *et al*. (2022).

Vê-se que a saúde, especialmente de cuidados preventivos e diagnósticos, terá um grande impulsionamento por conta de respostas mais rápidas advindas do emprego de novas tecnologias aplicadas à saúde, como o emprego de algoritmos inteligentes, como será visto a seguir.

2.1 As Ciências do Movimento Humano e um Futuro Digital

No universo humano contemporâneo, nunca se pensou tanto em recrutar máquinas no afã de dar celeridade a processos que outrora eram morosos. De acordo com Ueda (2022) nos próximos anos, algoritmos e Aprendizagem de Máquina (*Machine Learning*), estarão disponíveis em nuvem conectando habilidades cognitivas entre RNAs, aumentando assim a capacidade de resolução de problemas.

As implantações de algoritmos tendem a conectar o Ser humano ao código de programação, proporcionando mais agilidade em sua vida diária e eficiência ao mundo do trabalho, ainda segundo Ueda (2022)

> Essa megatendência *(sic)* será impulsionada pela convergência da conectividade global de alta conectividade de banda, como 5G, redes neurais e computação em nuvem. Ou seja, não apenas educação, vários outros setores serão impactados e transformados, como agronegócio, saúde e entretenimento. (p.1)

O futuro digital almejado por muitos, já é pragmatizado em grande parte da sociedade atual que busca para si a melhorias em seu estilo de vida, segundo Vilela Júnior (2020, p. 01) “todas estas tecnologias já existem e estão passando por processos de amadurecimento em diferentes níveis. Futurista demais? Não, são tecnologias em pleno desenvolvimento.”

Nesta sequência, surge a IA e a programação de algoritmos inteligentes para dar suporte com maior celeridade e eficácia que caminha a passos largos, apontando um cenário de muita relevância nessa área, tais como, corroborado por (GENEZINI, 2022; ALAUDDIN; BAHARUDDIN; MOHD GHAZALI, 2021; PASSOS, 2019).

Posto que a IA nos últimos anos tem acelerado o processo de digitalização mundial de negócios e serviços, principalmente o da saúde, agindo de

maneira disruptiva para uma transformação digital (ALAUDDIN; BAHARUDDIN; MOHD GHAZALI, 2021; LAKHANI; SUNDARAM, 2017).

Todavia, a aplicação de IA no objeto de estudo aqui apresentado, perpassa pelo desenvolvimento da RNA, muito especificamente a CNN que tratam do reconhecimento e identificação de características por meio de imagens (ALAUDDIN; BAHARUDDIN; MOHD GHAZALI, 2021; LAKHANI; SUNDARAM, 2017), que neste caso a CNN será aplicada na identificação de imagens de RX, já bastante abrangida no campo científico, identificando inclusive diversos tipos de câncer entre outras patologias, tais como as técnicas utilizadas nos trabalhos de Silva (2020), Baka *et al*. (2017) Belharbi *et al* (2017), um exemplo das várias possibilidades aplicações da visão computacional na área da saúde.

A seguir, serão caracterizados os materiais e métodos utilizados.

3 MATERIAIS E MÉTODOS

Esta é uma pesquisa descritiva de caráter pré-diagnóstico e métodos de IA aplicados na análise do movimento humano, para desenvolver e validar um algoritmo inteligente para a identificação de desvios da coluna lombar a partir de imagens de Raio X.

3.1 Caracterização da Amostra

A caracterização da amostra deu-se a partir de uma base de dados que foi elaborada sistematicamente com um grupo de pacientes previamente selecionados intencionalmente sem desvios de coluna lombar e outros diagnosticados com escoliose na mesma região, valendo-se inicialmente de 1200 imagens de raio x, cedidas por Instituto de Diagnóstico por Imagem da cidade de Curitiba estado do Paraná e mais 1697 imagens de dois bancos de dados de uso compartilhado são eles: o https://radiologyassistant.nl/ e https://radiopaedia.org/, pois neles tem-se estudos de casos reais e as imagens são cedidas livremente para pesquisa científicas, totalizando, nesta etapa, 2897 imagens. Estas imagens, na sequência, passaram por um processo de interpolação que faz pequenas alterações nas mesmas, consolidando um conjunto maior de imagens a serem utilizadas na CNN, integralizando 579400 diferentes imagens.

Foram utilizados nesta pesquisa, técnicas de visão computacional e inteligência artificial, para identificar imagens, especificamente as ferramentas utilizadas para criação de um algoritmo que partiram de um ambiente de desenvolvimento integrado, ou seja, IDE, *Spyder*® (versão 5.0.2) para criação e execução dos códigos, assim como, *PyCharm* (versão 2021.1.3), tudo isso, estruturado no código escrito na linguagem *Python*® (versão 3.9) em ambiente virtualizado denominado *Conda*®.

Tendo como bibliotecas base para a aplicação os pacotes: *TensorFlow* (2.6.0rc2), *OpenCV*® (*Python*) (4.5.3.56), *Numpy*® (1.21.1), Pandas® (1.3.1), *Matplotlib*® (3.4.2), além de bibliotecas embutidas do núcleo *Python* atualizadas de acordo com a versão 3.9.6 da linguagem.

Os dados gerados por meio de classificação multiclasse multinomial,

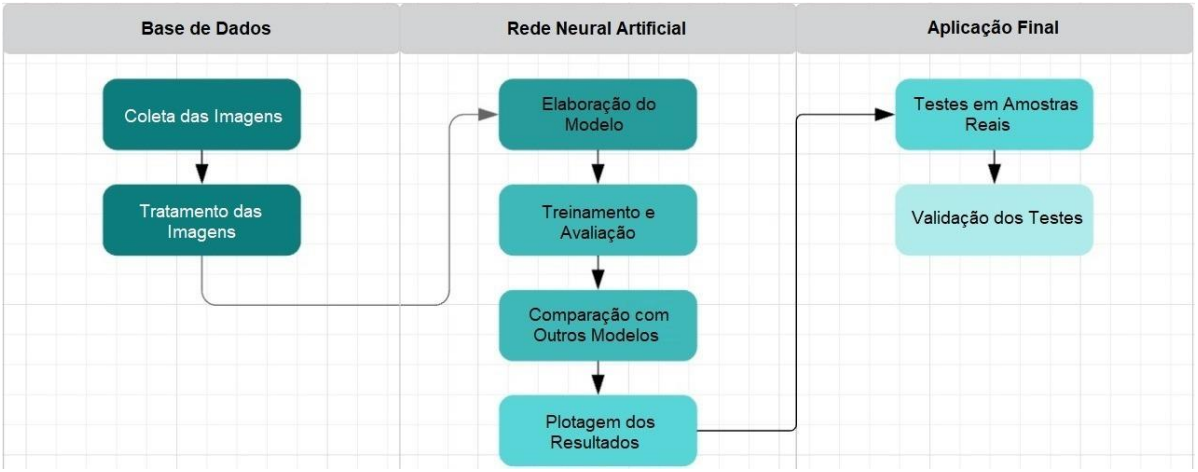
definida como regressão logística é considerados um dos métodos mais populares métodos discriminativos para problemas de classificação multiclasse. (LI; ZHU; DONG, 2018).

Esta pesquisa foi aprovada a partir do projeto integrado sob o título: *Métodos da Inteligência Artificial Aplicados na Análise do Movimento Humano,* aprovado, CAAE: 33912120.9.0000.5507 com parecer consubstanciado do Comitê de Ética e Pesquisa - CEP: 4126546, fonte primária e norteador deste estudo.

4 DESENVOLVIMENTO

A partir desse tópico será ilustrado todo o desenvolvimento do algoritmo proposto, a seguir na figura 1, será apresentado as etapas de construção do mesmo, bem como suas fases de maneira simplificada e simples visualização.

Figura 1 - Etapas de desenvolvimento do algoritmo.



Fonte: Autor, 2022

Contudo, profusos detalhamentos foram necessários, destacando-se todo o processo de construção do algoritmo que seguiu das seguintes etapas:

Preparação do ambiente de desenvolvimento

- Instalação das dependências
- Importação das bibliotecas, módulos e pacotes

Preparação da base de dados

- Carregamento da base de dados

- Pré-processamento das imagens
- Aumento físico da base de dados
- Tratamento das imagens
- Aplicação de filtros para detecção e extração de características
- Segmentação por binarização para identificação das máscaras
- Divisão da base em dados para treino e para teste
- Geração de mais imagens a partir da base inicial
- Aumento sintético das imagens a partir de suas matrizes

Criação da arquitetura da RNA

- Definição de modelo de rede neural
- Criação das camadas de convolução, polimento e normalização
- Criação das camadas de entrada, intermediárias e de saída
- Compilação

Treinamento e teste da RNA

- Alimentação da rede neural e início de processamento
- Avaliação do modelo
- Salvando o modelo

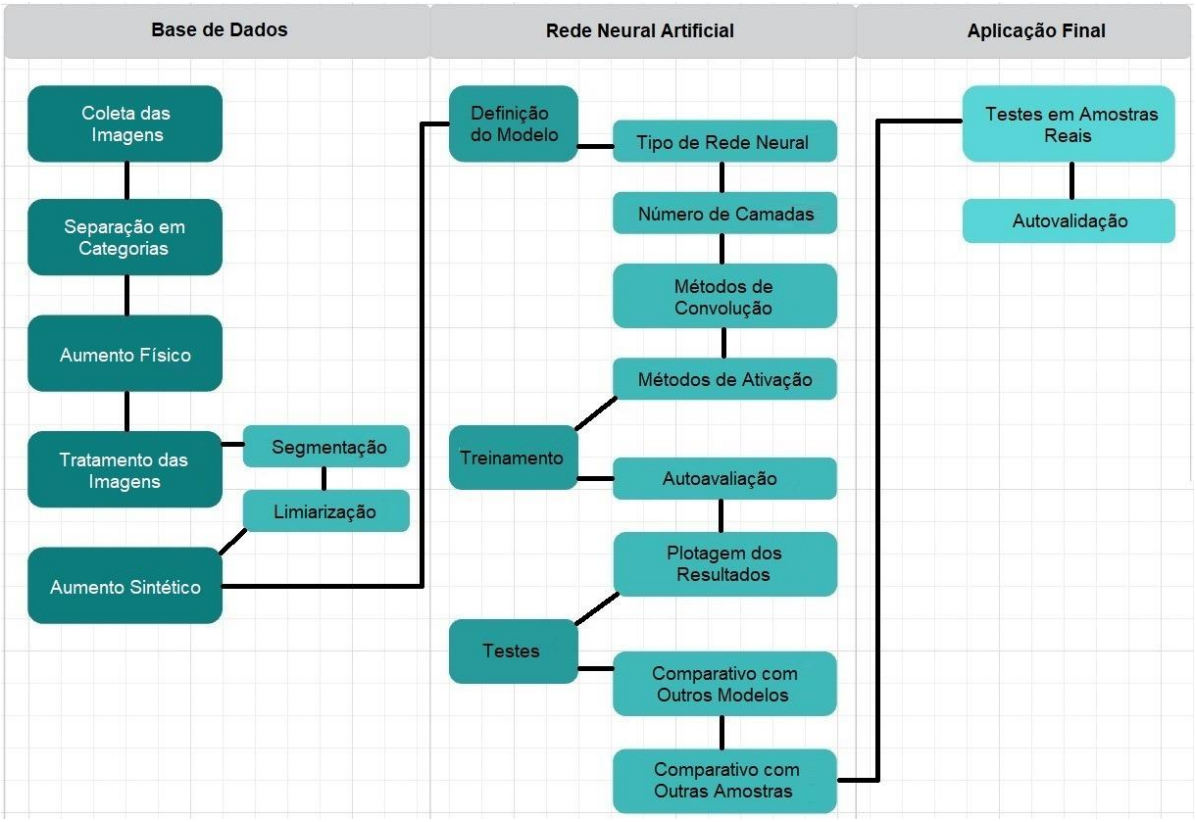
Testes em outros modelos, com diferentes parâmetros e configurações

Testes em amostras reais

Autovalidação

Para tanto, foi evidenciado as rotinas internas e suas ligações passo a passo no fluxograma abaixo**:**

Figura 2 - Etapas de detalhadas do algoritmo



Fonte: Autor, 2022.

A partir disso, criou-se um ambiente de trabalho que será visto a seguir no próximo tópico.

4.1 Ambiente de Desenvolvimento e Trabalho

Este estudo foi gerado em um ambiente virtualizado, no qual, foram

realizados diversos testes no melhor emprego das ferramentas utilizadas, extraindo as diversas possibilidades das bibliotecas disponíveis para a linguagem *Python*®, principalmente quanto existem diferentes nuances e configurações dos modelos de RNA, evitando assim, possíveis erros futuros de incompatibilidade em outras plataformas.

4.2 Instalação das Dependências

Nesta primeira etapa da preparação do ambiente de desenvolvimento, foram realizadas as devidas instalações das bibliotecas, módulos e pacotes que serão utilizadas ao longo do código.

Neste caso, boa parte das ferramentas as quais foram instrumentalizadas não vem por padrão instaladas no núcleo da linguagem *Python*®, logo, foi necessário instalar as mesmas, em um ambiente virtualizado.

Sendo assim, foram instaladas as bibliotecas *Numpy*®, *Pillow*®, *Matplotlib*®, *TensorFlow*®, *Keras*®, *Glob*® e *OpenCV*®. Ainda se tratando de compatibilidade, todos os processos foram criados valendo-se dos recursos dessas bibliotecas, criando ferramentas para a extração.

4.3 Importação das Bibliotecas, Módulos e Pacotes

Figura 3 - Importação de bibliotecas

```
import numpy as np
import PIL.Image
import matplotlib.pyplot as plt
from tensorflow import keras
from keras import backend as B
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.layers import LayerNormalization
from tensorflow.keras.layers import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.preprocessing import image
from keras.callbacks import LearningRateScheduler
from tensorflow.keras.optimizers import RMSprop, Adam
from keras.callbacks import ReduceLROnPlateau
from keras.utils.vis_utils import plot_model
from keras.layers import Conv2D, Conv2DTranspose
import keras.layers.convolutional as conv
import glob
import cv2

```

Fonte: Autor, 2022.

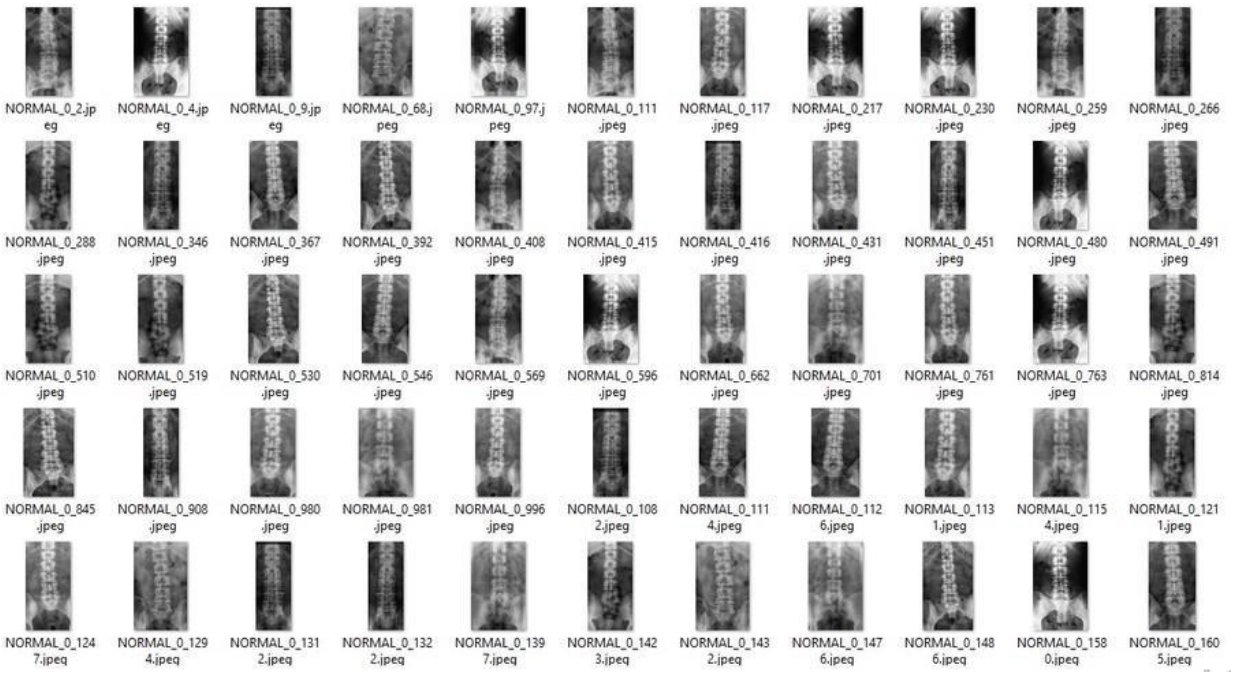
A critério de performance computacional de bibliotecas, módulos e pacotes utilizados como referência foram importadas apenas as ferramentas e que de fato eram necessárias, descartando assim, as ferramentas não utilizadas como vê-se na figura 3.

Uma vez instaladas todas as dependências, não havendo nenhum erro no processo de instalação, pode-se importá-las e dar início a compilação dos códigos pretendidos.

4.4 Tratamento da Base de Dados

Como este estudo se baseia em identificar patologias diretamente a partir de radiografias digitais de coluna, optou-se pelo desenvolvimento de um modelo de RNA, chamado convolucional, a CNN, que em outras palavras, possui sua arquitetura voltada a leitura e processamento de imagens, como reforça Abdou (2022), Kaur, Bhattacharya e Chana (2022) e Asri *et al*. (2022) que reconhece padrões a partir das mesmas, aprendendo de acordo com as características relevantes das amostras da base de dados, o que é um RX “normal”, assim como, um exame que apresenta alguma patologia, mais especificamente uma escoliose lombar.

Figura 4 - Base de dados



Fonte: Autor, 2022.

Para fins de compreensão do constructo o desenvolvimento partiu de uma base de dados inicial de 2897 imagens, um processo importante a ser aplicado é o chamado aumento físico da base de dados, onde foram geradas novas imagens a partir de uma base original, com pequenas variações morfológicas, visando não somente aumentar o volume/quantidade de imagens, mas obter na mesma base múltiplas imagens com características importantes mais facilmente identificáveis

para a RNA.

A base de dados parcial, após processamento dos mecanismos que se tem na sequência foi composta de 22880 imagens, sendo imagens de radiografias de coluna lombar em plano anatômico ântero-posterior, contendo no mesmo grupo imagens sem processamento, imagens com características destacadas (realce de bordas e contornos), imagens com pontos de marcação (inseridos manualmente) e imagens binarizadas segmentadas, destacando apenas os pontos de interesse, por sua vez separadas em uma base para treino e outra para teste e por fim, classificadas como normais ou patológicas.

Para este estudo, foi adotado, entre outras formas, a validação realizada a partir da própria base de dados, de modo que da base de dados final, já processada, parte da mesma foi utilizada para treino da rede neural, enquanto outra parte totalmente distinta, foi utilizada para teste e avaliação dos modelos.

4.5 Aumento Físico da Base de Dados

O código é desenvolvido retratado na figura 5 com um mecanismo que interpreta e carrega todas as imagens de uma determinada pasta, processando cada uma delas por uma ferramenta chamada Augmentor®, que por sua vez a partir de uma imagem fonte, gera novas imagens interpoladas com deformidades elásticas, de inclinação, rotação, zoom e espelhamento sem que se percam as características mais relevantes da imagem original, tudo configurado pelo próprio desenvolvedor, em outras palavras o Augmentor® realiza o fornecimento dos dados preditivos quantificados a partir de dados de comunicação de criptografia online.

Figura 5 - Importação do Augmentor

```
import Augmentor

p = Augmentor.Pipeline("dataset/")

p.random_distortion(probability=1, grid_width=2, grid_height=2, magnitude=5)
p.shear(probability=1, max_shear_left=1, max_shear_right=1)
p.rotate(probability=1, max_left_rotation=6, max_right_rotation=6)
p.zoom(probability=0.5, min_factor=1.0, max_factor=1.1)
p.flip_left_right(probability=0.4)

p.sample(200, multi_threaded=True)
p.process()

```

Fonte: Autor, 2022

Sintetizando, para cada imagem original, inicialmente foram geradas 200 novas imagens, ou seja, um total de 579400 imagens com características distintas.

4.6 Segmentação por Binarização / Limiarização

Quando se menciona redes neurais artificiais convolucionais que são aquelas dedicadas ao processamento a partir de imagens, uma prática muito comum surge ao combinar diferentes técnicas de aprendizado de máquina, com ferramentas externas que automatizam certos processos de tratamento de tais imagens.

Neste contexto, quando a base de dados é bruta, originalmente sem nenhum tipo de tratamento ou marcação, aplica-se os chamados mecanismos de convolução para extrair características destas imagens tentando contornar suas limitações apresentadas, entendendo-se que o mapeamento das imagens, pixel a pixel, extraindo características importantes das mesmas que ajudarão no processo de identificação de padrões por parte da RNA.

No entanto, a partir do momento em que se tem uma base de dados tratada com marcadores/máscaras nas regiões de interesse, pode-se partir para algo mais estruturado, combinando novas técnicas como Binarização, ou seja dupla de rotular cada pixel e Limiarização por Segmentação de contornos, as quais serão

úteis para extração das marcações específicas e aumento da base de dados via Redes Adversariais Generativas (GAN), ou seja, são aquelas que realizam a geração de novas imagens a partir dos marcadores mapeados anteriormente, em imagens de referência.

O processo de segmentação por sua vez é realizado com base em ferramentas de visão computacional, que é configurado todo um mecanismo para visualizar, apenas as marcações das imagens e reconhecer padrões a partir das mesmas. A geração de imagens a partir de GAN, é realizada por meio de arquitetura de rede neural que constitui uma nova base de dados com imagens geradas, com uma base de dados fonte, utilizando-se de alguns mecanismos discriminantes, para que a nova base de dados contenha apenas imagens válidas, com as mesmas características das originais.

Ao combinar diversas técnicas de IA, tais como, aplicação, mapeamento de marcações, geração de imagens por aumento físico, geração de imagens por aumento sintético, geração de imagens por redes GAN, obtém-se uma base de dados bastante robusta, que certamente irá contribuir para uma precisão, mais confiável margem de precisão nas classificações geradas a partir da rede neural.

Figura 6 - Entrada da binarização e limiarização

```
base_treino = '/dataset/train/normal/*'
lista_imagens = glob.glob(base_treino)

def exibe_imagem(imagem, titulo, tamanho):
  fig, axis = plt.subplots(figsize = tamanho)
  axis.imshow(imagem, 'gray')
  axis.set_title(titulo, fontdict = {'fontsize':22, 'fontweight':'medium'})
  plt.show()

imagem = lista_imagens[2]

img = cv2.imread(imagem)
thresh, img_thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

exibe_imagem(img, 'IMAGEM ORIGINAL', (5,5))
exibe_imagem(img_thresh, 'THRESH_BINARY', (5,5))

```

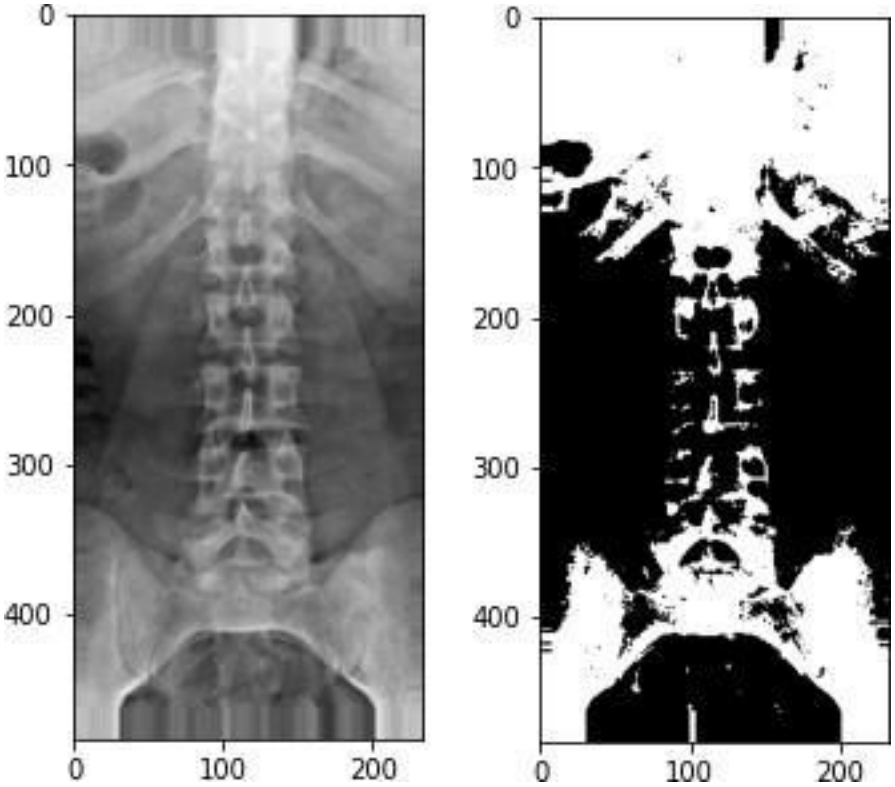
Fonte: Autor, 2022.

Para aplicar mecanismos de segmentação por binarização nas imagens, será aplicado a ferramenta *threshold* da biblioteca *OpenCV*®, por meio desta ferramenta define-se, então um limiar, ou seja, um valor de referência, que a partir deste, isola-se estruturas da imagem de acordo com seu valor de pixel, que se tratando de imagem radiográfica, diz respeito a densidade dos tecidos.

Na prática tal ferramenta recebe uma imagem, um valor referência de contorno, um valor de intervalo de tons de cinza e a técnica a ser aplicada, neste caso, binarização.

Quando se menciona valor referência, aplica-se a parametrização da função *threshold()* com o numeral 127, se define que todos os pixels com valor abaixo de 127 que serão convertidos para preto, assim como, todos os pixels de valor superior a 127 se tornarão totalmente brancos.

Figura 7 - Binarização e limiarização na imagem



Fonte: Autor, 2022.

Observando a imagem pôde-se observar que o valor de limiarização aplicado sobre a imagem original não produziu um resultado tão útil. Isto se dá,

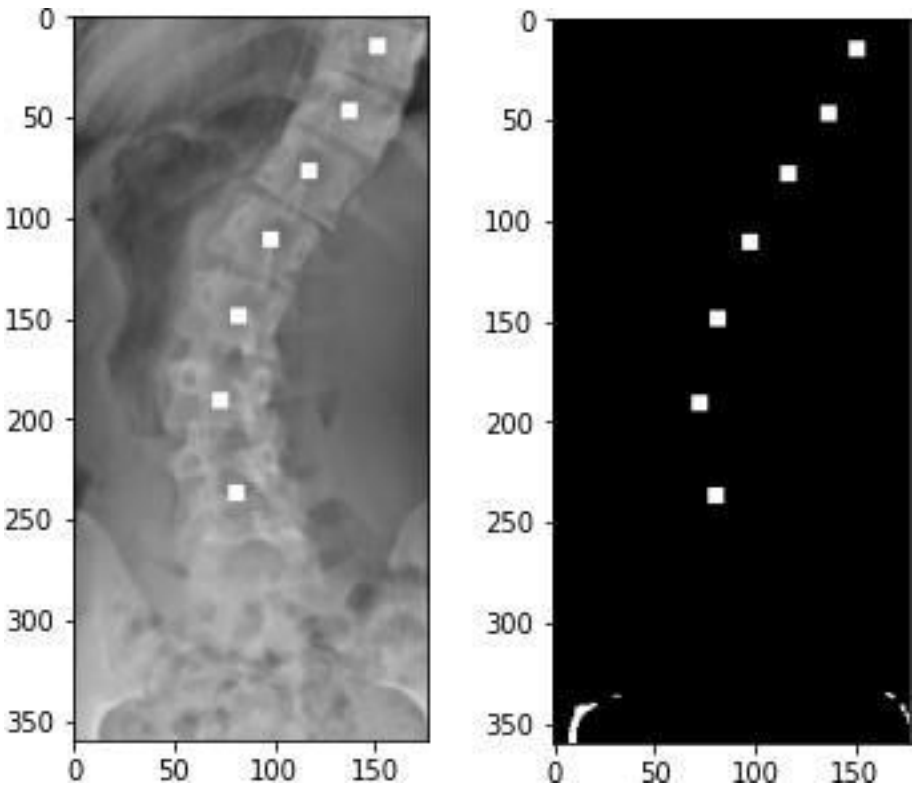
por se tratar de imagens de RX, cada tecido da anatomia corporal, possui uma densidade distinta, bem como, um respectivo valor de pixel, além disso, existem ainda muitas estruturas de densidades muito próximas, o que dificulta o processo de limiarização quando aplicado a imagens brutas, sem nenhum tipo de máscara ou marcação.

Figura 8 - Entrada com imagens marcadas

```
imagem = 'PATOLOGICA(6748).jpg'

img = cv2.imread(imagem)
thresh, img_thresh = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY)

exibe_imagem(img, 'IMAGEM ORIGINAL', (5,5))
exibe_imagem(img_thresh, 'THRESH_BINARY', (5,5))

```

Fonte: Autor, 2022.

Figura 9 - Binarização e limiarização com imagem demarcada



Fonte: Autor, 2022.

Já quando se aplica a mesma técnica sobre uma imagem com marcações, a limiarização ocorre como esperado, descartando todas as estruturas que não são de interesse, mantendo apenas as marcações. Desta forma, foi obtido um mapa de referência que ajudará a RNA a reconhecer padrões, neste caso, quando a rede neural realizar a leitura da imagem original, irá visualizar em conjunto a imagem binarizada para associar os pontos de marcação com o que se pretende identificar na imagem.

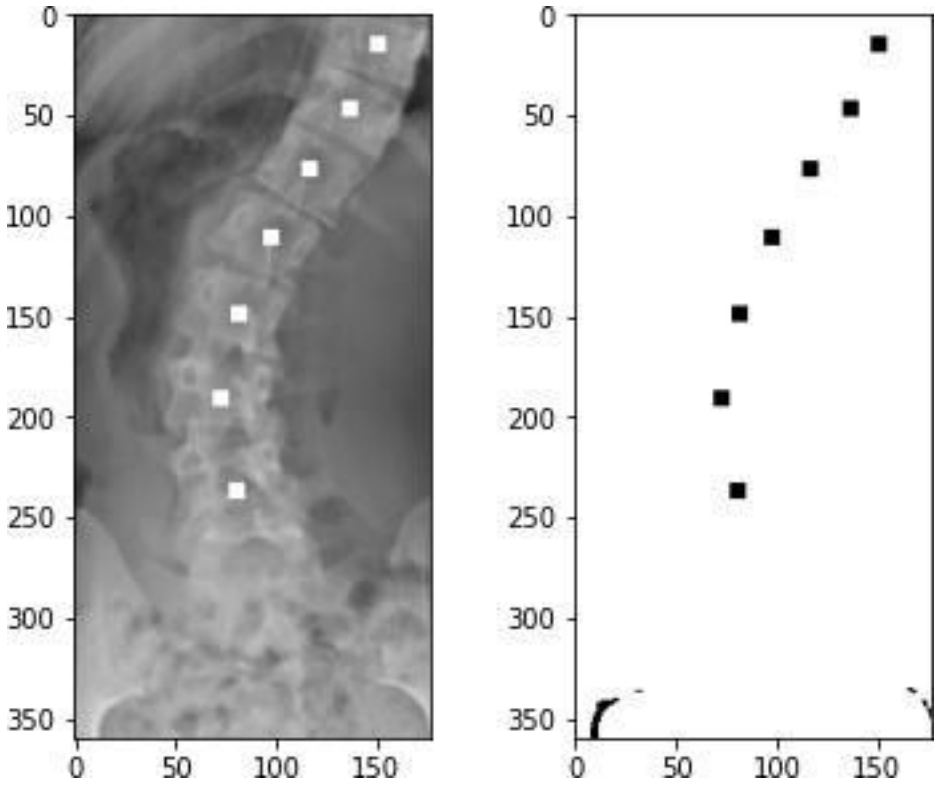
Ressalta-se que este estudo buscou treinar uma CNN para que identificasse padrões de curvatura da coluna lombar, logo, as marcações inseridas sobre os processos espinhosos das vértebras, foram a referência para que a RNA aprenda o que é uma curvatura patológica, frisando que a determinação de patologia se deu, por meio e imagens de RX já efetivamente laudadas como tais.

Figura 10 - Entrada de código invertendo P/B

```
1  imagem = 'PATOLOGICA(6748).jpg'
2
3  img = cv2.imread(imagem)
4  thresh, img_thresh = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY_INV)
5
6  exibe_imagem(img, 'Imagem Original', (5,5))
7  exibe_imagem(img_thresh, 'THRESH_BINARY_INV', (5,5))
8
```

Fonte: Autor, 2022

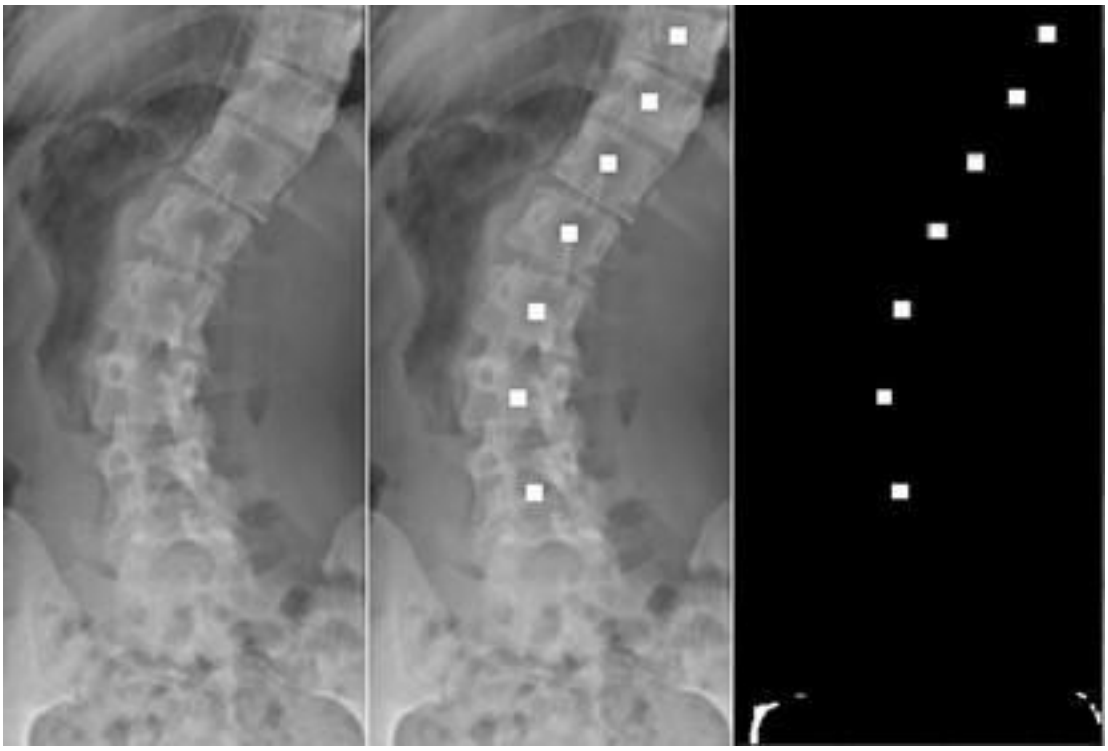
Figura 11 - Imagem demarcada invertida



Fonte: Autor, 2022.

Depois de realizar diversos testes, com diferentes configurações, foi possível chegar a um valor ideal de limiarização para a base de dados, o suficiente para extrair as características do eixo da coluna vertebral, sendo uma das informações a serem repassadas para a RNA em seu treinamento.

Figura 12 - Imagens limiarizadas com e sem marcação



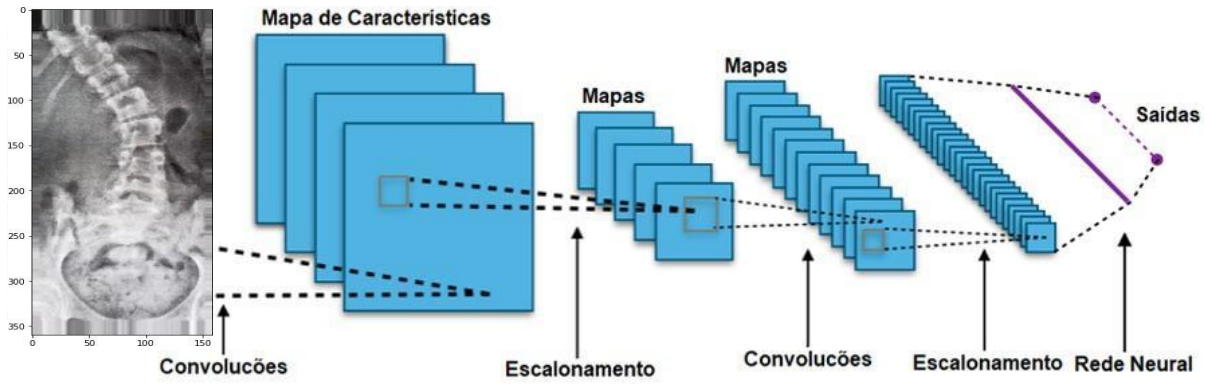
Fonte: Autor, 2022

Por fim, aplicou-se o processo de limiarização para todas as imagens da base de dados original, o qual obteve-se uma base de dados híbrida, composta de imagens não marcadas, imagens marcadas e imagens binarizadas, ou seja, três referências para a mesma amostra a serem identificadas e mapeadas pela RNA em busca de suas características relevantes.

Lembrando que em estudos radiológicos, as curvaturas da coluna compreendem diferentes variações da anatomia, sendo, curvaturas dentro de um limite normal e curvaturas que de acordo com seu nível de acentuação são julgadas como patológicas, pois não apenas podem ser o gatilho para lesões ósseas, mas também determinadas curvaturas, indicam compressão de partes moles adjacentes.

4.7 Mecanismo de Convolução

Figura 13 - Mecanismo de convolução



Fonte: Autor, 2022

Como mecanismo de convolução têm-se o mecanismo de conversão de uma imagem para uma matriz numérica, onde cada pixel da composição da mesma é mapeado com um valor de acordo com suas características.

Uma imagem de RX possui diferentes valores comuns associados a diferentes valores de densidades de tecidos da anatomia, tudo isso representado dentro de um intervalo numérico entre 0 e 255 em uma escala que vai desde o preto com a menor densidade possível, até o branco com maior densidade possível, como a porção cortical de um osso. Desse modo, a partir de uma imagem de RX pode-se visualizar a mais fidedigna anatomia de um determinado paciente que será utilizada para fins diagnósticos, tendo em vista que grande parte das patologias identificáveis a partir deste tipo de exame tende a gerar uma imagem específica de diferença de densidade.

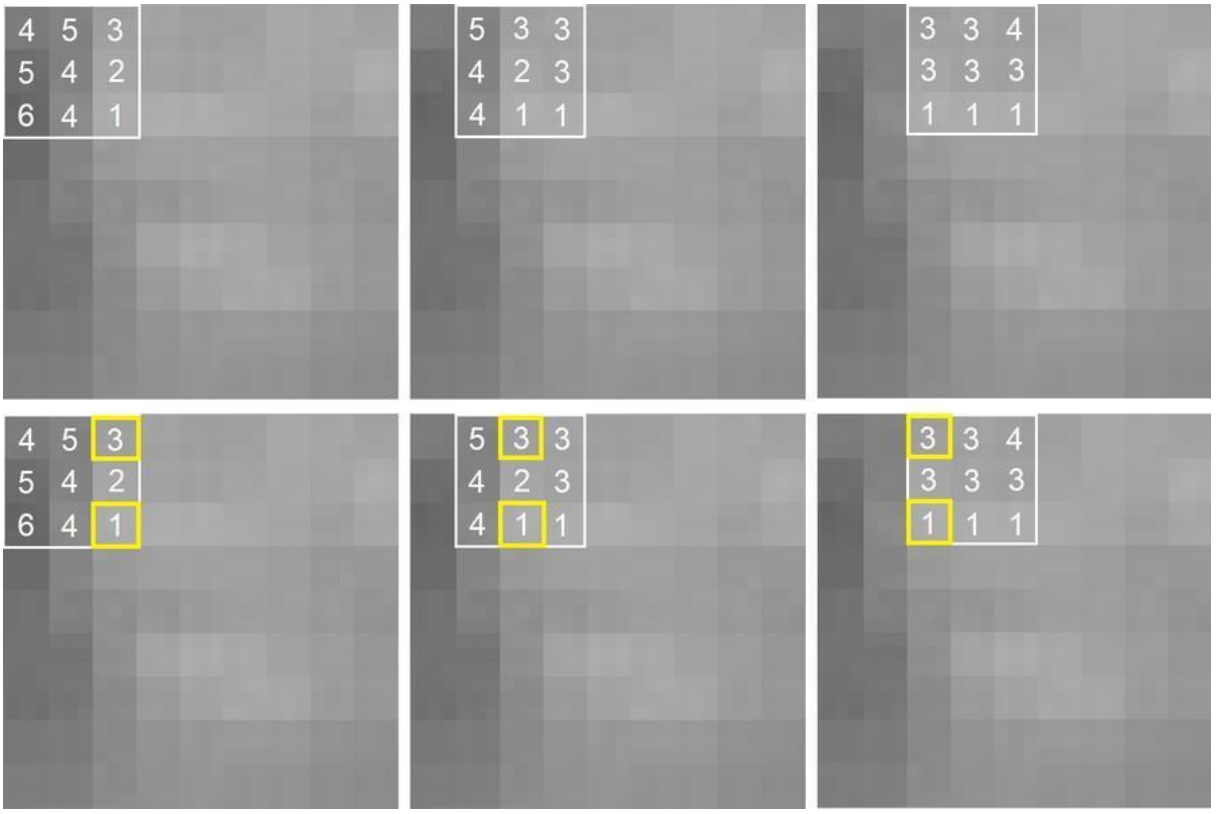
Em outras palavras, a partir de um exame de imagem o médico radiologista busca diferenças de densidades dos tecidos que podem indicar patologias instaladas sobre um determinado órgão ou tecido.

Na visão computacional, pode-se realizar uma série de mapeamentos e processamentos em cada pixel ou grupo de pixels da mesma para então, gerar uma matriz numérica que por sua vez alimentará uma arquitetura de RNA para que sejam reconhecidos os devidos padrões das características daquela imagem.

Esse processo é realizado por uma série de ferramentas que irão realizar

uma varredura na imagem, pixel a pixel, linha por linha, bloco por bloco, mapeando características e salvando em mapas que posteriormente terão seus valores utilizados como neurônios da camada de entrada de uma RNA.

Figura 14 - Processamentos em cada pixel



Fonte: Autor, 2022

Neste processo foi realizado a extração manual de características das imagens, pois, para este estudo, que parte de imagens de RX de coluna, com o desafio de conseguir em meio ao mecanismo de convolução, identificar e mapear características da anatomia como posição da coluna, bordas e contornos dos corpos das vértebras, espaços entre vértebras e ainda de tirar o foco de estruturas, tais como ossos da bacia e da caixa torácica que neste caso permanecem dificultando uma melhor visualização.

O fato é que isso tornou-se possível, combinando ferramentas das bibliotecas *OpenCV*® e *Numpy*®, onde foi realizado a leitura da imagem e o mapeamento da mesma, gerando matrizes para blocos de pixels.

Como exemplo de mapeamento, no primeiro quadrante foi aplicada uma matriz de pixels de dimensão 3x3 e na sequencia realizado a leitura do valor de cada um dos pixels e a partir desta informação, a matriz se move para direita mapeando outro quadrante, comparando os valores do quadrante anterior com o atual, aplicando uma série de métricas para quando considerar uma grande alteração de pixels entre os quadrantes utilizados para comparação, salvando essa característica em uma nova matriz que chamada de mapa.

Cada filtro possui suas próprias métricas de como determinar uma possível diferenciação de pixels relevantes e suficientes para considerar que naquele determinado ponto da imagem existe alguma característica visual relevante de algum objeto.

Figura 15 - Entrada do código de filtros

```
image1_BGR = cv2.imread('AP.0001.jpg')
image1_GRAY = cv2.cvtColor(image1_BGR, cv2.COLOR_BGR2GRAY)
image11_GRAY = image1_BGR[:, :, 2] * 0.299 + \
               image1_BGR[:, :, 1] * 0.587 + \
               image1_BGR[:, :, 0] * 0.114
image11_GRAY = np.around(image11_GRAY, decimals=0).astype(np.int)
image1_GRAY_pad = np.pad(image1_GRAY, (1, 1), mode='constant', constant_values=0)

filter_1 = np.array([[1, 0, -1],
                     [2, 0, -2],
                     [1, 0, -1]])
filter_2 = np.array([[1, 2, 1],
                     [0, 0, 0],
                     [-1, -2, -1]])
filter_3 = np.array([[1, 0, -1],
                     [1, 0, -1],
                     [1, 0, -1]])

output_image1_GRAY = np.zeros(tuple([3]) + image1_GRAY.shape)

for i in tqdm(range(image1_GRAY_pad.shape[0] - 2)):
    for j in range(image1_GRAY_pad.shape[1] - 2):
        patch = image1_GRAY_pad[i:i+3, j:j+3]
        output_image1_GRAY[0, i, j] = np.sum(patch * filter_1)
        output_image1_GRAY[1, i, j] = np.sum(patch * filter_2)
        output_image1_GRAY[2, i, j] = np.sum(patch * filter_3)

output_image1_GRAY = np.clip(output_image1_GRAY, 0, 115)

```

Fonte: Autor, 2022.

O código acima foi desenvolvido a partir de um mecanismo que carrega uma imagem fonte da base de dados, convertendo a mesma para uma escala de cinza padrão, em seguida, é extraído grupos de 9x9 pixels com valores mapeados. Lembrando que no processo de varredura da imagem, anterior eram apenas identificados os pixels, atribuindo valores para os mesmos e agora com o processo de aplicação de filtros para cada pixel da imagem são realizadas algumas métricas de comparação entre seu valor, o valor de pixel anterior, o valor de pixel posterior, assim como de todas as adjacências dentro do grupo 9x9. Dessa forma, apenas as características importantes por grupo de pixel serão mapeadas e indexadas como referência.

Já no processo de varredura aplicando filtros sobre a imagem, quando aplicado filtro por filtro em cada imagem, os valores dos pixels pré-definidos no grupo 9x9 do filtro são multiplicados pelos valores mapeados anteriormente de cada grupo 9x9 da imagem original, resultando em um novo mapa 9x9 com características realçadas. Vale ressaltar que este processo foi aplicado a cada pixel de cada imagem, em todas as imagens da base de dados.

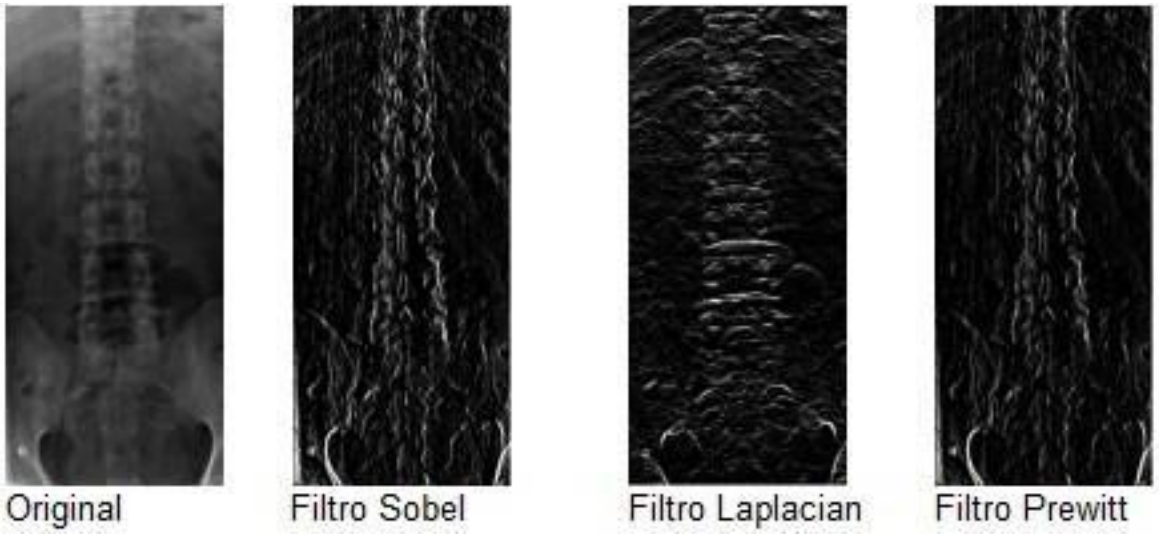
Figura 16 - Entrada do código de filtros sobre a imagem

```
plt.rcParams['figure.figsize'] = (10.0, 14.0)
figure, ax = plt.subplots(nrows=3, ncols=4)
ax[0, 0].imshow(image1_GRAY, cmap=plt.get_cmap('gray'))
ax[0, 1].imshow(output_image1_GRAY[0], cmap=plt.get_cmap('gray'))
ax[0, 2].imshow(output_image1_GRAY[1], cmap=plt.get_cmap('gray'))
ax[0, 3].imshow(output_image1_GRAY[2], cmap=plt.get_cmap('gray'))
ax[0, 0].set_title('Imagem Original', fontsize=16)
ax[0, 1].set_title('Filtro Sobel', fontsize=16)
ax[0, 2].set_title('Filtro Laplacian', fontsize=16)
ax[0, 3].set_title('Filtro Prewitt', fontsize=16)
for i in range(3):
    for j in range(4):
        ax[i, j].axis('off')
plt.tight_layout()
plt.subplots_adjust(left=0.1, right=0.9,
                    bottom=0.1, top=0.9,
                    wspace=0.1, hspace=0.1)

figure.savefig('/imagem.png')
figure.canvas.set_window_title('Imagens com realce de caracteristicas')

plt.show()

```

Fonte: Autor, 2022.

Em seguida foi criada uma estrutura mais simplificada para plotar as imagens lado a lado para facilitar os resultados individuais das aplicações de filtros, bem como, sua visualização.

Figura 17 - Imagens com filtros



Fonte: Autor, 2022.

Dessa forma, obteve-se um resultado interessante, onde cada tipo de filtro individualmente evidencia um resultado bastante particular ao de tentar realçar características das estruturas da coluna lombar em si. Depois de fortalecida a base de dados com o processamento dos filtros *Sobel, Lapalacian e Prewitt* combinados, focando nas matrizes das imagens, pode favorecer a RNA, a identificar e reconhecer tais características específicas, aprimorando-a ainda mais todo o constructo.

4.8 Aumento sintético da Base de Dados

Finalizando a etapa de tratamento da base de dados, um último recurso válido para este propósito é o chamado aumento sintético da base de dados, o qual é elaborado somente sobre as matrizes das imagens, em outras palavras, somente sobre os valores numéricos mapeados podendo gerar novas imagens de características parecidas e similares.

Figura 18 - Aumento sintético da base de dados

```
gerador_treino = ImageDataGenerator(rescale = 1.0/255,
                                    rotation_range = 5,
                                    horizontal_flip = True,
                                    shear_range = 0.2,
                                    height_shift_range = 0.07,
                                    zoom_range = 0.2)
gerador_teste = ImageDataGenerator(rescale = 1.0/255)

base_treino = gerador_treino.flow_from_directory('dataset/train/',
                                                 target_size = (128,128),
                                                 batch_size = 16,
                                                 class_mode = 'binary')
base_teste = gerador_teste.flow_from_directory('dataset/test/',
                                               target_size = (128,128),
                                               batch_size = 16,
                                               class_mode = 'binary')

```

Fonte: Autor, 2022

Para este processo utilizou-se a ferramenta *ImageDataGenerator*, que irá ler todas as imagens de um determinado diretório, convertendo os valores das matrizes para o intervalo de tons de cinza, em seguida escalonando as imagens para o tamanho 128x128 pixels, classificando as mesmas de forma binária, lembrando que para cada base têm-se imagens separadas em normal e patológica.

4.9 Redes Neurais Convolucionais – CNN (*Convolutional Neural Networks*)

4.9.1 Classificador Base

Como classificador base, foi criado um modelo de CNN apenas com as camadas essenciais para o funcionamento do modelo.

Figura 19 - Entrada classificador base

```
classificador = Sequential()
classificador.add(Conv2D(128,
                         kernel_size = (3,3),
                         input_shape = (128,128,3),
                         activation = 'relu'))
classificador.add(Flatten())
classificador.add(Dense(units = 128,
                        activation = 'relu'))
classificador.add(Dense(units = 1,
                        activation = 'sigmoid'))
classificador.compile(optimizer = 'Adam',
                      loss = 'binary_crossentropy',
                      metrics = ['accuracy'])

```

Fonte: Autor, 2022.

Sendo assim, o modelo foi composto apenas com uma camada convolucional, seguido de uma camada planificadora e uma camada densa com sua respectiva camada classificadora.

Na sequência foi configurado o compilador, neste caso baseado em entropia cruzada binária utilizando como métrica a maior acurácia possível.

Figura 20 – Compilador

```
epochs = 100

learning_rate = ReduceLROnPlateau()

h_0 = classificador.fit(base_treino,
                        steps_per_epoch = 100,
                        epochs = epochs,
                        validation_data = base_teste,
                        callbacks = [learning_rate],
                        verbose = 1,
                        validation_steps = 25)

```

Fonte: Autor, 2022.

Contudo, definiu-se o número de vezes que a CNN foi processar os

dados, em seguida alimentando a mesma com os dados das imagens da base de treino, usando de 100 em 100 amostras, validando os mesmos com dados das imagens da base de teste, empregando como parâmetro de retorno o mecanismo de taxa de aprendizado baseado em *ReduceLROnPlateau,* como se definiu anteriormente. Portanto, esta métrica aponta para o mecanismo de descida dos gradientes, o modo como os melhores valores encontrados são salvos e aplicados para cada novo ciclo de processamento.

Figura 21 - Taxa de aprendizado épocas iniciais

```
Epoch 1/100
100/100 [==============================] - 101s 841ms/step - loss: 7.3738 - accuracy: 0.6200 - val_loss: 0.6281 - val_accuracy: 0.5600
Epoch 2/100
100/100 [==============================] - 78s 779ms/step - loss: 1.4548 - accuracy: 0.6200 - val_loss: 0.5588 - val_accuracy: 0.6400
Epoch 3/100
100/100 [==============================] - 77s 768ms/step - loss: 1.0087 - accuracy: 0.7300 - val_loss: 0.5044 - val_accuracy: 0.6400
Epoch 4/100
100/100 [==============================] - 74s 740ms/step - loss: 0.5795 - accuracy: 0.7800 - val_loss: 0.4989 - val_accuracy: 0.7200
Epoch 5/100
100/100 [==============================] - 73s 726ms/step - loss: 0.9770 - accuracy: 0.8200 - val_loss: 0.5199 - val_accuracy: 0.6400
Epoch 6/100
100/100 [==============================] - 65s 659ms/step - loss: 0.4737 - accuracy: 0.8400 - val_loss: 0.2536 - val_accuracy: 0.8400
Epoch 7/100
100/100 [==============================] - 61s 619ms/step - loss: 0.4726 - accuracy: 0.8500 - val_loss: 0.4324 - val_accuracy: 0.6800
Epoch 8/100
100/100 [==============================] - 59s 587ms/step - loss: 0.5595 - accuracy: 0.8300 - val_loss: 7.4295 - val_accuracy: 0.5600
Epoch 9/100
100/100 [==============================] - 63s 624ms/step - loss: 0.9514 - accuracy: 0.7200 - val_loss: 0.3991 - val_accuracy: 0.7600
Epoch 10/100
100/100 [==============================] - 63s 626ms/step - loss: 0.4168 - accuracy: 0.8100 - val_loss: 4.0362 - val_accuracy: 0.6400
```

Fonte: Autor, 2022.

Ao executar o bloco de código anterior mencionado na figura 21 é possível acompanhar via terminal, o processamento da rede, que começa apresentando como retorno uma enorme disparidade, consequentemente um valor de acurácia muito baixo e um valor de taxa de perda muito acentuado.

Desta forma, a CNN não consegue encontrar nenhuma combinação entre os padrões encontrados na base de treino para a base de teste. Após algumas épocas de processamento os primeiros padrões começam a ser de fato encontrados e validados, mesmo que com uma margem de acerto muito baixa.

Figura 22 - Taxa de aprendizado épocas finais

```
Epoch 90/100
100/100 [==============================] - 4s 43ms/step - loss: 0.1974 - accuracy: 0.9000 - val_loss: 1.0210 - val_accuracy: 0.6800
Epoch 91/100
100/100 [==============================] - 5s 48ms/step - loss: 0.1840 - accuracy: 0.9200 - val_loss: 0.2944 - val_accuracy: 0.8400
Epoch 92/100
100/100 [==============================] - 5s 50ms/step - loss: 0.1217 - accuracy: 0.9600 - val_loss: 0.9947 - val_accuracy: 0.6400
Epoch 93/100
100/100 [==============================] - 4s 43ms/step - loss: 0.0894 - accuracy: 0.9800 - val_loss: 0.5342 - val_accuracy: 0.8000
Epoch 94/100
100/100 [==============================] - 6s 57ms/step - loss: 0.1403 - accuracy: 0.9700 - val_loss: 0.7075 - val_accuracy: 0.8000
Epoch 95/100
100/100 [==============================] - 4s 37ms/step - loss: 0.1690 - accuracy: 0.9400 - val_loss: 0.5409 - val_accuracy: 0.7600
Epoch 96/100
100/100 [==============================] - 4s 36ms/step - loss: 0.1012 - accuracy: 0.9600 - val_loss: 0.7435 - val_accuracy: 0.7600
Epoch 97/100
100/100 [==============================] - 4s 42ms/step - loss: 0.1245 - accuracy: 0.9600 - val_loss: 1.2797 - val_accuracy: 0.6400
Epoch 98/100
100/100 [==============================] - 5s 50ms/step - loss: 0.1555 - accuracy: 0.9400 - val_loss: 0.5668 - val_accuracy: 0.7600
Epoch 99/100
100/100 [==============================] - 3s 30ms/step - loss: 0.2133 - accuracy: 0.9100 - val_loss: 0.5627 - val_accuracy: 0.8000
Epoch 100/100
100/100 [==============================] - 4s 43ms/step - loss: 0.1342 - accuracy: 0.9200 - val_loss: 0.3749 - val_accuracy: 0.9200
Precisão do treino: 98.00%
Precisão da validacao: 92.00%
```

Fonte: Autor, 2022.

Ao final do processamento do modelo, vê-se que são retornados valores muitos altos em relação a precisão, todavia ainda não fidedignos, uma vez que se processou, apenas a base de dados inicial com um modelo considerado básico e de teste.

Observando os valores retornados das últimas épocas de processamento é possível notar uma grande oscilação entre os mesmos, o que na prática reflete que tais dados não são confiáveis.

A CNN deste modelo básico, pode até retornar valores altos de precisão inicial em suas métricas de avaliação, todavia o que se observa é que dos grupos de amostras processadas, possui uma margem entre 10% a 20% das mesmas são perdidas, números estes que são ainda piores nas métricas de validação, o que não reflete em um resultado final prático de 98% de precisão como apontado.

Tudo isto ocorre, pois, a CNN em si possui métricas de autoavaliação que por padrão retornam os melhores valores para suas margens de acerto.

Contudo, em um olhar mais crítico, pode-se identificar que tais valores não se mantêm em constância, logo, não são confiáveis, pelo menos preliminarmente.

Figura 23 - Sumário do Classificador Base

```
1  classificador.summary()
2
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 128)     3584
_________________________________________________________________
flatten (Flatten)            (None, 2032128)           0
_________________________________________________________________
dense (Dense)                (None, 128)               260112512
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 260,116,225
Trainable params: 260,116,225
Non-trainable params: 0
```

Fonte: Autor, 2022

Analisando o sumário gerado para este modelo de CNN, pode-se observar que apesar do pequeno número de camadas, o modelo em sua arquitetura foi moldado de modo a realizar cerca de 260 milhões de parâmetros. Isto reflete a grande dificuldade para encontrar os padrões corretos e replicá-los para outras instâncias da CNN.

É como se a rede aprendesse o que é característica de uma imagem de RX normal e de uma imagem de RX patológica, porém tivesse que escolher uma entre 260 milhões que coincidisse em 98% com as características de seu aprendizado para então, poder realizar sua classificação.

Figura 24 - Classificação *Keras*

```
keras2ascii(classificador)

        OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

            Input   #####    128  128    3
           Conv2D    \|/  -------------------      3584     0.0%
             relu   #####    126  126  128
          Flatten   ||||| -------------------         0     0.0%
                    #####     2032128
            Dense   XXXXX ------------------- 260112512   100.0%
             relu   #####          128
            Dense   XXXXX -------------------       129     0.0%
          sigmoid   #####            1
```

Fonte: Autor, 2022.

Ao analisar a projeção de distribuição dos nós da CNN, conforme figura 24, observa-se que a mesma se rearranjou de modo muito desbalanceado, onde 99% das informações dos padrões de características foram encontrados apenas nos *perceptrons* da rede, e o restante a partir das imagens. Isso reflete que os padrões encontrados podem ser qualquer tipo de padrão, por simples aproximação e que podem não representar de fato características importantes das imagens, até porque boa parte das mesmas é composta de informações visuais irrelevantes para seu propósito de ênfase nas estruturas da coluna lombar.

Figura 25 - Histórico de processamento

```
print('Precisão do treino: {0:.2f}%'.format(max(h_0.history['accuracy']) * 100))
print('Precisão da validacao: {0:.2f}%'.format(max(h_0.history['val_accuracy']) * 100))

Precisão do treino: 98.00%
Precisão da validacao: 92.00%
```

Fonte: Autor, 2022.

Ao término do processamento da CNN, automaticamente a mesma gera um histórico, vide figura 23, o qual, foi extraído o último valor referente a suas margens de acerto dos processos de treino e de teste. Embora, como dito no tópico anterior, este modelo ao que tudo indica, não refletem a realidade do modelo.

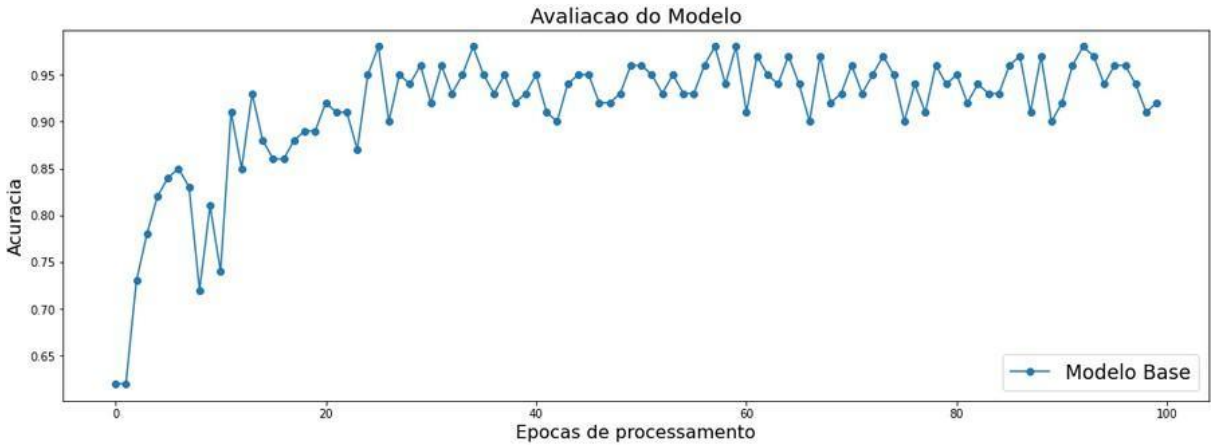
Figura 26 - Entrada de plotagem

```
plt.rcParams['figure.figsize'] = (18.0, 6.0)
plt.plot(h_0.history['accuracy'], '-o')
plt.legend(['Modelo Base'], loc = 'lower right', fontsize = 'xx-large')
plt.xlabel('Epocas de processamento', fontsize=16)
plt.ylabel('Acuracia', fontsize=16)
plt.title('Avaliacao do Modelo', fontsize=18)
plt.show()

```

Fonte: Autor, 2022.

Todavia, plotou-se alguns gráficos baseados nos dados retornados pelo modelo, para que de forma mais intuitiva possa se verificar a evolução do modelo durante toda sua execução.

Gráfico 1 - Acurácia em função dos números de épocas - classificador base



Fonte: Autor, 2022.

Do histórico gerado após o término do processamento da rede, pode-se plotar do campo *accuracy*, de acordo com o gráfico 1, salientando os valores de acurácia do modelo ao longo das 100 épocas de execução do modelo.

Confirmando o que foi mencionado anteriormente, apesar do modelo logo nas épocas iniciais começar a encontrar padrões, a instabilidade dos mesmos se mantém ao longo de todas as épocas, hora melhorando, hora piorando, totalmente instável, não condizente com o valor retornado.

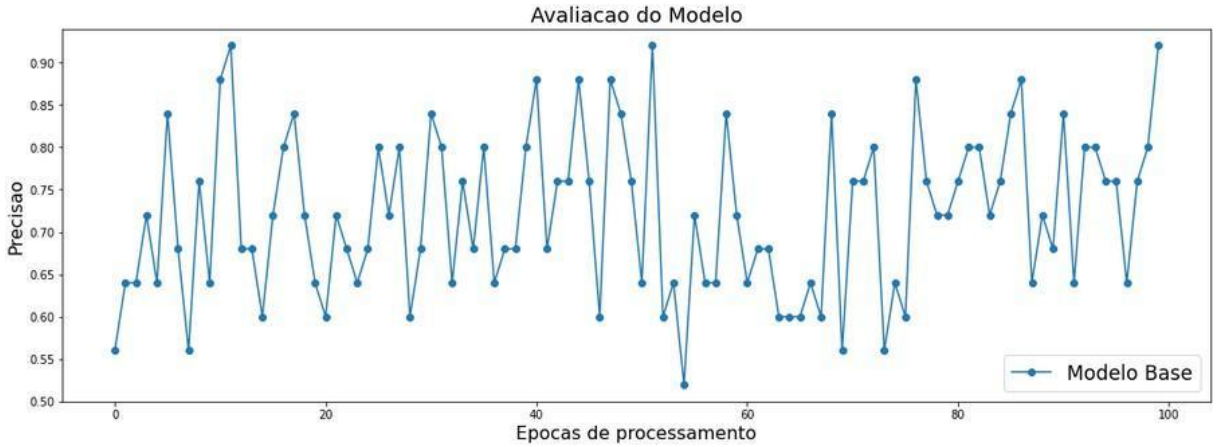
Figura 27 - Entrada da validação da acurácia

```
plt.rcParams['figure.figsize'] = (18.0, 6.0)
plt.plot(h_0.history['val_accuracy'], '-o')
plt.legend(['Modelo Base'], loc = 'lower right', fontsize = 'xx-large')
plt.xlabel('Epocas de processamento', fontsize=16)
plt.ylabel('Precisao', fontsize=16)
plt.title('Avaliacao do Modelo', fontsize=18)
plt.show()

```

Fonte: Autor, 2022.

Da mesma forma, plotando as informações salvas no campo *val_accuracy,* pode-se notar que o processo de validação da acurácia, lembrando que este processo nada mais é do que o mesmo processo de validação normal, porém aplicado sobre a base de dados de teste.

***Gráfico 2* – Precisão em função do número de épocas - classificador base**



Fonte: Autor, 2022.

Neste momento destaca-se uma instabilidade ainda maior, ao longo de todas as épocas de processamento, o que apenas confirma a baixa confiabilidade dos dados e seus valores retornados, tanto a partir da base de treino quanto da base de teste.

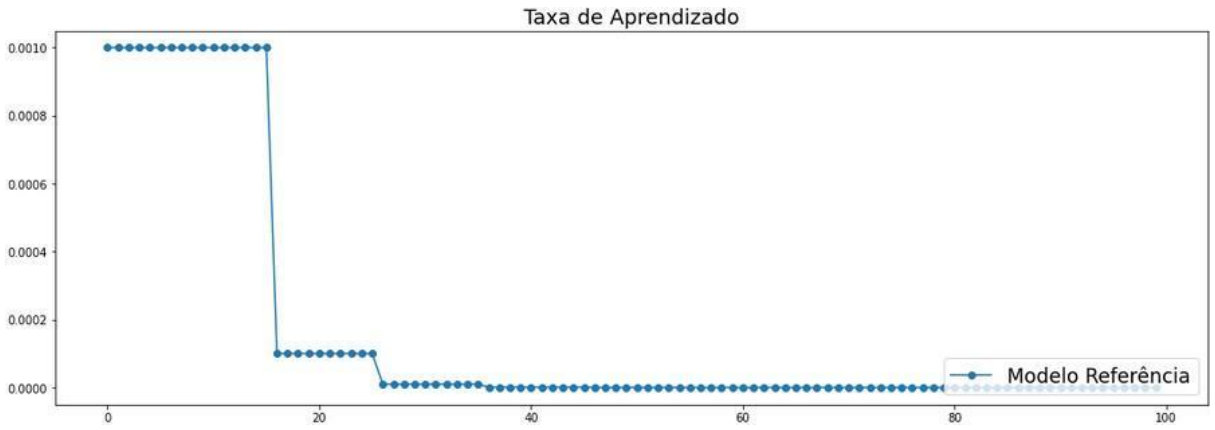
Figura 28 - Histórico da taxa de aprendizado

```
plt.rcParams['figure.figsize'] = (18.0, 6.0)
plt.plot(h_0.history['lr'], '-o')
plt.legend(['Modelo Referência'], loc = 'lower right', fontsize = 'xx-large')
plt.title('Taxa de Aprendizado', fontsize=18)
plt.show
```

Fonte: Autor, 2022.

Plotando os dados referentes ao campo ‘lr’ apresentam-se os dados referentes ao histórico da taxa de aprendizado da rede neural.

Gráfico 3 - Aprendizado em função do número de épocas - classificador base



Fonte: Autor, 2022.

Neste ponto inicial a rede não havia encontrado nenhum possível padrão a ser replicado e em torno da época 20 foi encontrado um valor de descida de gradiente, que é posteriormente corrigido mais duas vezes e mantido até o final do processamento, de modo que a rede neural havia julgado ter encontrado um ótimo valor, todavia sabe-se que tal valor de taxa de aprendizado refletiu em uma rede neural instável.

4.10 Classificador Referência

Como classificador “referência” utilizou-se o modelo de classificador

usado para a programação anterior, referente ao modelo de melhor eficiência para classificação da base de dados sem marcações. O desafio para este modelo atual é a partir de uma base de dados diferente, produzindo um modelo tão eficiente ou ainda mais eficiente que o modelo referência já testado anteriormente.

É importante ressaltar que este mesmo modelo, parte de suas configurações originais e será posto a teste para a nova base de dados, justamente para verificar seu comportamento com a mesma.

Figura 29 - Entrada do classificador referência

```
classificador = Sequential()
classificador.add(Conv2D(128,
                         kernel_size = (3,3),
                         input_shape = (128,128,3),
                         activation = 'relu'))
classificador.add(BatchNormalization())
classificador.add(MaxPooling2D(pool_size = (2,2)))
classificador.add(Conv2D(32,
                         (3,3),
                         activation = 'relu'))
classificador.add(BatchNormalization())
classificador.add(MaxPooling2D(pool_size = (2,2)))
classificador.add(Flatten())
classificador.add(Dense(units = 128,
                        activation = 'relu'))
classificador.add(Dense(units = 128,
                        activation = 'relu'))
classificador.add(Dense(units = 1,
                        activation = 'sigmoid'))
optimizer = RMSprop(lr=0.001,
                    rho=0.9,
                    epsilon=1e-08,
                    decay=0.0)
classificador.compile(optimizer = optimizer,
                      loss = 'binary_crossentropy',
                      metrics = ['accuracy'])

```

Fonte: Autor, 2022.

Em referência a arquitetura deste modelo, evidencia-se que o mesmo possui duas camadas de convolução com suas respectivas camadas

intermediárias de normalização e polimento, assim como três camadas densas sendo duas de *perceptrons* e uma final de classificação binária.

Portanto, para esse modelo que passa a ser considerado otimizador, foi implementada a ferramenta *RMSprop* que por sua vez foi parametrizada com algumas configurações de ajuste fino para buscar o melhor valor possível de descida dos gradientes.

Figura 30 - Entrada de novos parâmetros finos de ajuste

```
28  epochs = 100
29
30  learning_rate = ReduceLROnPlateau(monitor='accuracy',
31                                    patience=3,
32                                    verbose=1,
33                                    factor=0.5,
34                                    min_lr=0.00001)
35
36  h_1 = classificador.fit(base_treino,
37                          steps_per_epoch = 100,
38                          epochs = epochs,
39                          validation_data = base_teste,
40                          callbacks = [learning_rate],
41                          verbose = 1,
42                          validation_steps = 25)
43
```

Fonte: Autor, 2022.

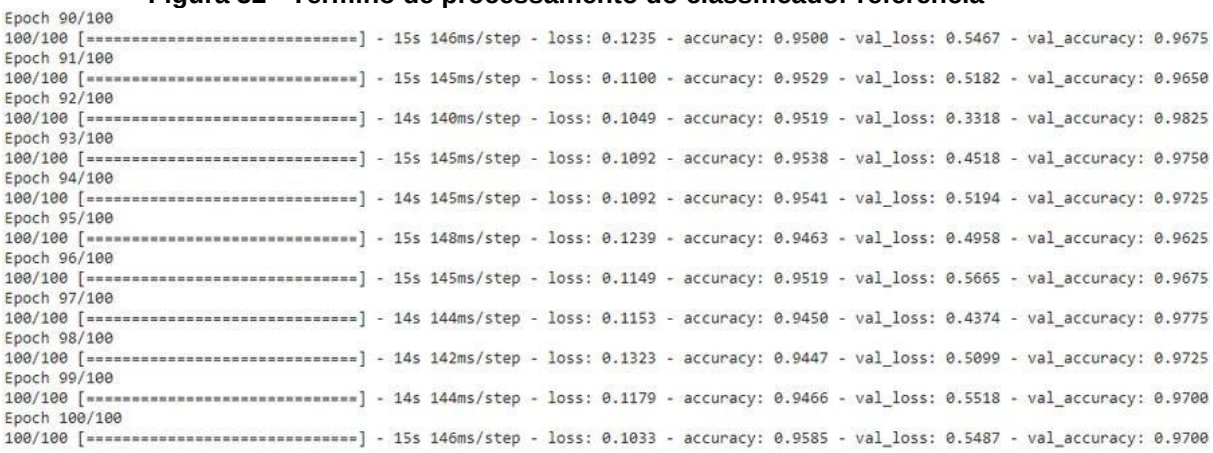
Por fim, é definido que o modelo que foi executado cem vezes, finalizando a CNN alimentada com os dados da base de treino, assim como uma série de configurações são definidas como ajuste fino para que no processo de aprendizado de máquina se obtenha os melhores valores em margens de acerto possíveis.

Figura 31 - Classificador de referência em execução

Fonte: Autor, 2022.

Colocando o modelo em execução, via terminal pode-se observar o progresso e como esperado, nas primeiras épocas de execução o modelo encontra uma certa dificuldade, até reconhecer seus primeiros padrões, todavia assim que encontra uma taxa de aprendizado com suficiência, o mesmo começa a replicar a mesma na rede, melhorando seus resultados e evoluindo época, após época de processamento.

Figura 32 - Término de processamento do classificador referência



Fonte: Autor, 2022.

Ao término do processamento da CNN é possível visualizar que diferentemente do modelo anterior, os dados retratam-se mais consistentes, segundo as métricas utilizadas pela própria rede neural para medir sua eficiência,

após processar a base de dados ao longo de 100 épocas, chegou-se em valores em acima dos 95% de acurácia, validados também acima dos 95% na fase de validação.

Deste modo, vê-se que tanto os testes aplicados sobre a base de dados de treino, assim como sobre a base de dados para teste, os valores de acurácia retornados dos mesmos são próximos e estáveis, podendo ser utilizados como referência, ou seja, um modelo viável de referência.

Figura 33 - Sumário do classificador referência

```
1  classificador.summary()
2
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 128)     3584
_________________________________________________________________
batch_normalization (BatchNo (None, 126, 126, 128)     512
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        36896
_________________________________________________________________
batch_normalization_1 (Batch (None, 61, 61, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 28800)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               3686528
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129
=================================================================
Total params: 3,744,289
Trainable params: 3,743,969
Non-trainable params: 320
```

Fonte: Autor, 2022

Observando o sumário da CNN nota-se que a partir deste modelo os dados foram “melhor” distribuídos entre as camadas de convolução e densas se comparado ao modelo base.

Nota-se que da maneira como os dados foram rearranjados em suas camadas, o modelo produziu em sua estrutura algo em torno de 3 milhões e 744 mil parâmetros, número também menor se comparado ao número encontrado no modelo base.

É importante ressaltar que um número muito alto de parâmetros, não necessariamente irá gerar um ótimo resultado de busca de padrões pela rede neural, pois um número muito grande de conexões a partir de uma base de dados pequena ou desbalanceada gera o chamado *overfitting* de acordo com Vilela Junior, *et al*. (2022), quando o modelo perde o controle de seus parâmetros, gerando padrões errôneos que acabam sendo replicados na rede afetando o seu aprendizado de máquina, o que não foi o caso encontrado.

Figura 34 - Aplicando o *Keras* no classificador referência

```
1   keras2ascii(classificador)
2
```

```
          OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

              Input   #####    128  128    3
             Conv2D    \|/  -------------------      3584     0.1%
               relu   #####    126  126  128
 BatchNormalization    μ|σ  -------------------       512     0.0%
                      #####    126  126  128
       MaxPooling2D   Y max -------------------         0     0.0%
                      #####     63   63  128
             Conv2D    \|/  -------------------     36896     1.0%
               relu   #####     61   61   32
 BatchNormalization    μ|σ  -------------------       128     0.0%
                      #####     61   61   32
       MaxPooling2D   Y max -------------------         0     0.0%
                      #####     30   30   32
            Flatten   ||||| -------------------         0     0.0%
                      #####       28800
              Dense   XXXXX -------------------   3686528    98.5%
               relu   #####         128
              Dense   XXXXX -------------------     16512     0.4%
               relu   #####         128
              Dense   XXXXX -------------------       129     0.0%
            sigmoid   #####           1
```

Fonte: Autor, 2022.

Analisando a distribuição dos nós da rede para assim deduzir suas possíveis interconexões, é possível notar que ainda grande parte dos dados estão concentrados nas camadas densas, todavia é proporcional ao número de parâmetros e por sua vez, possuindo um volume mais bem atribuído às camadas de convolução, possivelmente um valor suficiente para o modelo de acordo com o tamanho da base de dados

Figura 35 - Entrada para impressão da precisão e validação

```
1  print('Precisão do treino: {0:.2f}%'.format(max(h_1.history['accuracy']) * 100))
2  print('Precisão da validacao: {0:.2f}%'.format(max(h_1.history['val_accuracy']) * 100))
3
```

```
Precisão do treino: 95.85%
Precisão da validacao: 99.50%
```

Fonte: Autor, 2022.

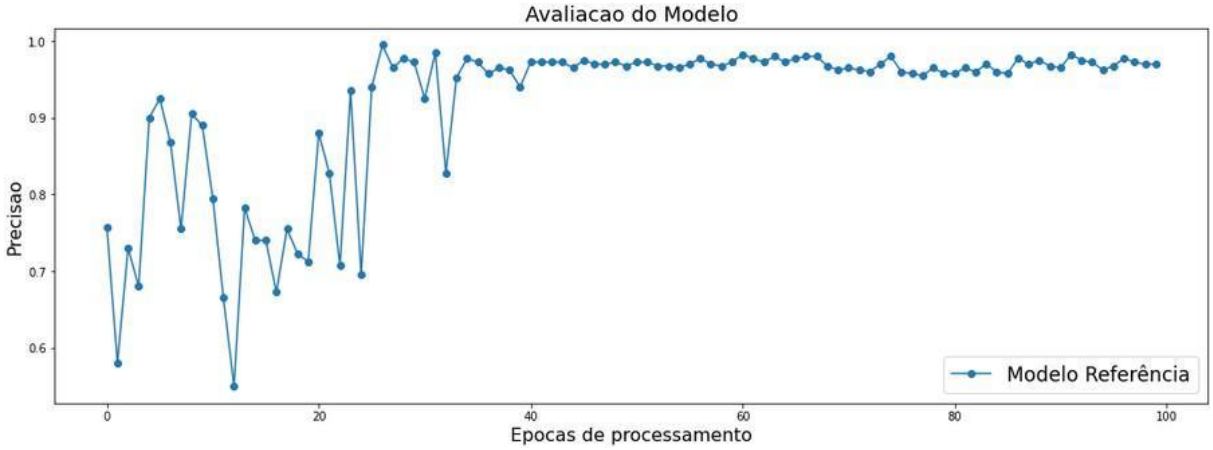
Como visto logo após o término do processamento da CNN concluídas as fases de treino e de teste da rede nos é retornado um valor acima de 95% de acurácia, confirmado por sua métrica de validação.

Gráfico 4 - Acurácia em função do número de épocas - classificador referência

Fonte: Autor, 2022.

Conforme o gráfico 4 referente a evolução da CNN, nota-se que mesmo no início das épocas houve uma crescente e exponencial variação positiva da acurácia do modelo, logo nas fases iniciais, o que não ocorreu outro no modelo básico, ou seja nas primeiras épocas de processamento já se conseguiu valores que rapidamente evoluíram ao longo de todas as épocas de execução, encontrando por volta da época 30 um valor expressivo, o qual foi replicado até o final do processo visando encontrar o melhor valor de acurácia possível.

Gráfico 5 - Precisão em função do número de épocas - classificador referência



Fonte: Autor, 2022.

Analisando o gráfico 5 referente à precisão, identifica-se que o modelo ao longo das primeiras 40 épocas de execução oscilou bastante até encontrar valores bons o suficiente para serem replicados. A partir do momento que tais valores de aprendizado de máquina foram encontrados, é possível notar que rapidamente os valores mensurados como precisão do modelo evoluíram e se mantiveram relativamente estáveis ao longo do restante do processamento.

Gráfico 6 - Taxa de aprendizado em função de épocas - classificador referência

Fonte: Autor, 2022.

Complementar ao gráfico 6, analisando a taxa de aprendizado da rede neural destaca-se que um valor próximo ao ideal esperado foi encontrado por volta da época 40 de processamento, sofrendo poucos reajustes até o término do

processo, sendo estes ajustes encontrando valores cada vez melhores.

4.11 Classificador X1

Uma vez realizados os testes dos modelos base e referência, vislumbrou-se a possibilidade a partir deste ponto a criação de novos modelos. Sendo estes mais robustos, com diferentes configurações e totalmente adaptados a base de dados para que no futuro busque modelos mais eficientes para o propósito final de identificação de imagens de RX em duas categorias distintas.

Figura 36 – Entrada do código Classificador X1

```
classificadorx1 = Sequential()
classificadorx1.add(Conv2D(128,
                           kernel_size = (6,6),
                           padding = 'same',
                           input_shape = (128,128,3),
                           activation = 'relu'))
classificadorx1.add(BatchNormalization())
classificadorx1.add(MaxPooling2D(pool_size = (2,2)))
classificadorx1.add(Flatten())
classificadorx1.add(Dense(units = 128,
                          activation = 'relu'))
classificadorx1.add(Dropout(0.2))
classificadorx1.add(Dense(units = 128,
                          activation = 'relu'))
classificadorx1.add(Dropout(0.2))
classificadorx1.add(Dense(units = 1,
                          activation = 'sigmoid'))
optimizer = RMSprop(learning_rate=0.001,
                    rho=0.9,
                    epsilon=1e-08,
                    decay=0.0)
classificadorx1.compile(optimizer = optimizer,
                        loss = 'binary_crossentropy',
                        metrics = ['accuracy'])

```

Fonte: Autor, 2022.

Com a arquitetura do modelo pretendido, o destaque fica por conta de

uma CNN de modelo sequencial *“fully connected”,* o qual é aplicado, a partir de uma base de dados processada sequencialmente por diversas camadas subsequentes e ao final do processo, métricas de padrões encontrados são aplicados em novas amostras. (SCHWING; URTASUN, 2015)

Neste modelo em particular, têm-se uma camada de convolução com algumas camadas intermediárias de filtragem das imagens em meio ao processo de convolução que na sequência apresentam-se três camadas densas, sendo duas de 128 neurônios cada e uma de 1 neurônio que é o nó referente a classificação. Define-se também, um mecanismo de otimização baseado na ferramenta *RMSprop*, com alguns ajustes finos definidos, visando realizar o processo de descida dos gradientes, seja o melhor possível.

Figura 37 - Entrada das taxas de aprendizado classificador X1

```
epochs = 100

learning_rate = ReduceLROnPlateau(monitor='accuracy',
                                  patience=3,
                                  verbose=1,
                                  factor=0.5,
                                  min_lr=0.00001)

h_x1 = classificadorx1.fit(base_treino,
                           steps_per_epoch = 100,
                           epochs = epochs,
                           validation_data = base_teste,
                           callbacks = [learning_rate],
                           verbose = 1,
                           validation_steps = 25)

```

Fonte: Autor, 2022

Do mesmo modo como empregado nos modelos anteriores, são definidas algumas configurações, por meio de códigos de programação, referentes a taxa de aprendizado, épocas de processamento e inicialização da função que coloca a CNN em teste.

Figura 38 - Taxas de aprendizado classificador X1 épocas iniciais

```
Epoch 1/100
100/100 [==============================] - 16s 149ms/step - loss: 4.5838 - accuracy: 0.6470 - val_loss: 5.3571 - val_accuracy: 0.5275
Epoch 2/100
100/100 [==============================] - 15s 152ms/step - loss: 1.1194 - accuracy: 0.7431 - val_loss: 0.3538 - val_accuracy: 0.8200
Epoch 3/100
100/100 [==============================] - 15s 147ms/step - loss: 0.8784 - accuracy: 0.7776 - val_loss: 8.2407 - val_accuracy: 0.5300
Epoch 4/100
100/100 [==============================] - 15s 152ms/step - loss: 0.9864 - accuracy: 0.7751 - val_loss: 17.2591 - val_accuracy: 0.5350
Epoch 5/100
100/100 [==============================] - 15s 151ms/step - loss: 1.0066 - accuracy: 0.7827 - val_loss: 2.0043 - val_accuracy: 0.7525
Epoch 6/100
100/100 [==============================] - 15s 149ms/step - loss: 0.7750 - accuracy: 0.8134 - val_loss: 12.1353 - val_accuracy: 0.7050
Epoch 7/100
100/100 [==============================] - 15s 150ms/step - loss: 1.2738 - accuracy: 0.7769 - val_loss: 3.8179 - val_accuracy: 0.8650
Epoch 8/100
100/100 [==============================] - 15s 149ms/step - loss: 0.5145 - accuracy: 0.8138 - val_loss: 179.0514 - val_accuracy: 0.5225
Epoch 9/100
100/100 [==============================] - 15s 152ms/step - loss: 0.5041 - accuracy: 0.8300 - val_loss: 52.3457 - val_accuracy: 0.5150
Epoch 10/100
100/100 [==============================] - 15s 150ms/step - loss: 0.5110 - accuracy: 0.8256 - val_loss: 21.7317 - val_accuracy: 0.4950
```

Fonte: Autor, 2022

Figura 39 - Taxas de aprendizado classificador X1 épocas finais

```
Epoch 90/100
100/100 [==============================] - 16s 156ms/step - loss: 0.1618 - accuracy: 0.9306 - val_loss: 0.0490 - val_accuracy: 0.9775
Epoch 91/100
100/100 [==============================] - 15s 151ms/step - loss: 0.1763 - accuracy: 0.9171 - val_loss: 0.0900 - val_accuracy: 0.9600
Epoch 92/100
100/100 [==============================] - 15s 154ms/step - loss: 0.1853 - accuracy: 0.9156 - val_loss: 0.0620 - val_accuracy: 0.9675
Epoch 93/100
100/100 [==============================] - 15s 150ms/step - loss: 0.1909 - accuracy: 0.9112 - val_loss: 0.0689 - val_accuracy: 0.9625
Epoch 94/100
100/100 [==============================] - 15s 152ms/step - loss: 0.1563 - accuracy: 0.9221 - val_loss: 0.1403 - val_accuracy: 0.9450
Epoch 95/100
100/100 [==============================] - 15s 148ms/step - loss: 0.1819 - accuracy: 0.9121 - val_loss: 0.0889 - val_accuracy: 0.9550
Epoch 96/100
100/100 [==============================] - 15s 153ms/step - loss: 0.1594 - accuracy: 0.9278 - val_loss: 0.1321 - val_accuracy: 0.9425
Epoch 97/100
100/100 [==============================] - 15s 154ms/step - loss: 0.1804 - accuracy: 0.9225 - val_loss: 0.0843 - val_accuracy: 0.9525
Epoch 98/100
100/100 [==============================] - 15s 151ms/step - loss: 0.1873 - accuracy: 0.9183 - val_loss: 0.0988 - val_accuracy: 0.9500
Epoch 99/100
100/100 [==============================] - 15s 151ms/step - loss: 0.1814 - accuracy: 0.9108 - val_loss: 0.1047 - val_accuracy: 0.9525
Epoch 100/100
100/100 [==============================] - 15s 151ms/step - loss: 0.1853 - accuracy: 0.9183 - val_loss: 0.1611 - val_accuracy: 0.9375
```

Fonte: Autor, 2022.

Nos mesmos moldes dos exemplos anteriores, acompanhando o processamento da CNN, identifica-se a cada época uma evolução, atingindo índices de acurácia superiores a 90%, estáveis, todavia com uma taxa de perda relativamente alta para este propósito. Por outro lado, ao refletir a cerca dos valores de *loss* que são as perdas, estes ficam em torno de 18%, significa que a cada 100 imagens processadas, por volta de 18 não foram possíveis de serem classificadas de acordo com os mesmos padrões que classificaram corretamente o restante das amostras.

Figura 40 - Sumário do classificador X1

```
1  classificadorx1.summary()
2
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 128, 128, 128)     13952
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 128, 128)     512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 128)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 524288)            0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               67108992
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 129
=================================================================
Total params: 67,140,097
Trainable params: 67,139,841
Non-trainable params: 256
```

Fonte: Autor, 2022.

Ao realizar uma breve análise do sumário da figura 40, vê-se que o mesmo entre suas camadas e os nós se auto-organizaram em torno de 67 milhões de parâmetros.

Figura 41 - Aplicando o *Keras* para o classificador X1

```
keras2ascii(classificadorx1)

          OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

              Input   #####    128  128    3
             Conv2D    \|/  -------------------     13952     0.0%
               relu   #####    128  128  128
 BatchNormalization    μ|σ  -------------------       512     0.0%
                      #####    128  128  128
       MaxPooling2D   Y max -------------------         0     0.0%
                      #####     64   64  128
            Flatten   ||||| -------------------         0     0.0%
                      #####      524288
              Dense   XXXXX -------------------  67108992   100.0%
               relu   #####          128
            Dropout    | || -------------------         0     0.0%
                      #####          128
              Dense   XXXXX -------------------     16512     0.0%
               relu   #####          128
            Dropout    | || -------------------         0     0.0%
                      #####          128
              Dense   XXXXX -------------------       129     0.0%
            sigmoid   #####            1
```

Fonte: Autor, 2022.

Novamente, em função de a rede neural ter se organizado com um volume exacerbado de parâmetros, a maior parte das mesmas acabou por se concentrar a nível das camadas densas, logo, é esperado uma inconsistência dos valores de aprendizado de máquina a serem retornados.

Figura 42 - Entrada para impressão da precisão e validação

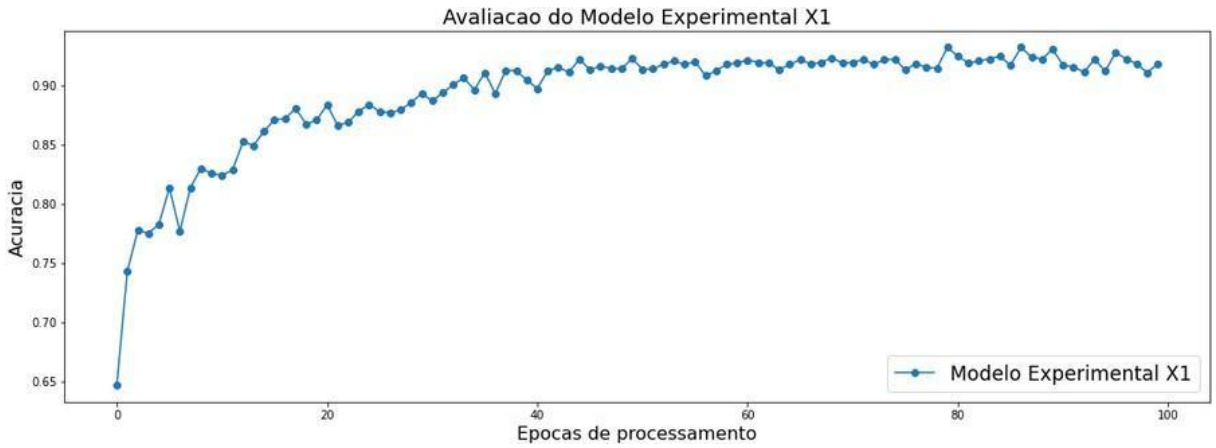
```
print('Precisão do treino: {0:.2f}%'.format(max(h_x1.history['accuracy']) * 100))
print('Precisão da validacao: {0:.2f}%'.format(max(h_x1.history['val_accuracy']) * 100))

Precisão do treino: 93.22%
Precisão da validacao: 100.00%
```

Fonte: Autor, 2022.

Exibindo em terminal os resultados das métricas de autoavaliação da

rede neural, a mesma apresenta uma acurácia em torno de 93%, o que até poderia ser condizente com o modelo, porém, de acordo com o resultado da validação, 100%, pode-se deduzir que de fato a acurácia isolada é um dado que pode ser inconsistente.

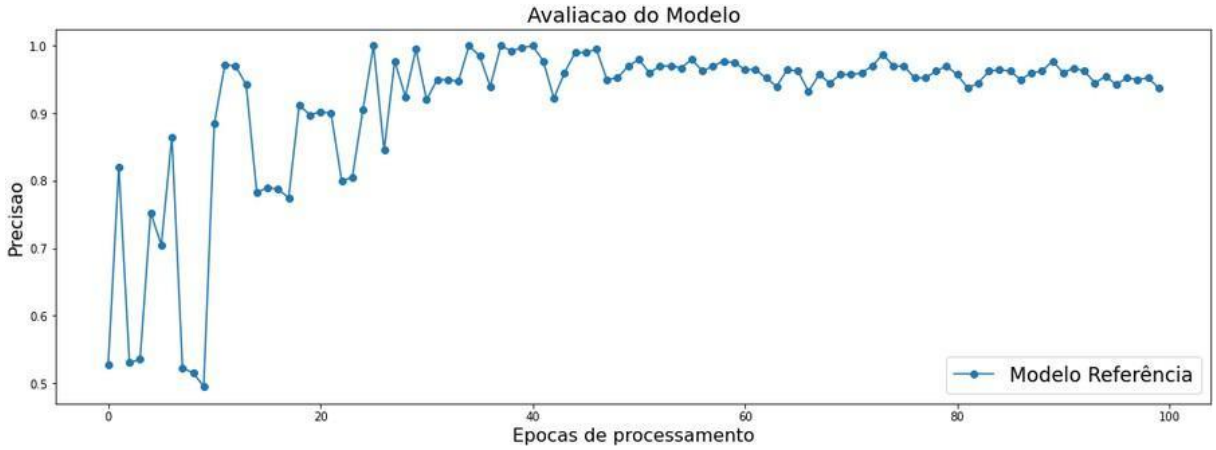
Gráfico 7 - Acurácia em função do número de épocas - classificador X1



Fonte: Autor, 2022

Analisando o gráfico 7, observa-se que a partir da época 50 a acurácia atinge um patamar com discretas flutuações.

Gráfico 8 - Precisão em função do número de épocas - classificador X1



Fonte: Autor, 2022.

Da mesma forma, o gráfico 8 referente a precisão, pode-se observar uma relativa constância a partir da metade das épocas de processamento.

Gráfico 9 - Aprendizado em função do número de épocas - classificador X1

Fonte: Autor, 2022.

Como esperado, de acordo com o gráfico 9, por volta da metade das épocas de processamento a rede neural encontrou o menor limiar de valores de taxa de aprendizado, ou seja, uma evidente estabilização do aprendizado de máquina.

O ponto a ser destacado aqui, é que o modelo em função de suas pequenas inconsistências, acabou por ser menos eficiente para seu propósito, haja visto que sua margem de acertos, em torno de 93%, é relativamente baixa se comparado a outros modelos de configurações diferentes.

4.12 Classificador X2

Em sequência a programação da rede neural, tem-se um novo modelo, nomeado apenas como X2, baseado no modelo X1, todavia com diferentes características por parte de sua arquitetura e configurações, adicionando mais camadas de processamento visando uma melhor distribuição dos dados nos neurônios da rede para assim encontrar melhores padrões, consequentemente uma melhor margem de acertos em classificação.

Figura 43 - Entrada do código classificador X2

```
classificadorx2 = Sequential()
classificadorx2.add(Conv2D(128,
                           kernel_size = (3,3),
                           padding = 'same',
                           input_shape = (128,128,3),
                           activation = 'relu'))
classificadorx2.add(BatchNormalization())
classificadorx2.add(MaxPooling2D(pool_size = (2,2)))
classificadorx2.add(Conv2D(32,
                           (3,3),
                           activation = 'relu'))
classificadorx2.add(BatchNormalization())
classificadorx2.add(MaxPooling2D(pool_size = (2,2)))
classificadorx2.add(Conv2D(32,
                           (3,3),
                           activation = 'relu'))
classificadorx2.add(BatchNormalization())
classificadorx2.add(MaxPooling2D(pool_size = (2,2)))
classificadorx2.add(Conv2D(32,
                           (3,3),
                           activation = 'relu'))
classificadorx2.add(BatchNormalization())
classificadorx2.add(MaxPooling2D(pool_size = (2,2)))
classificadorx2.add(Flatten())
classificadorx2.add(Dense(units = 128,
                           activation = 'relu',
                           use_bias=True,
                           kernel_initializer='glorot_uniform',
                           bias_initializer='zeros'))
classificadorx2.add(Dropout(0.2))
classificadorx2.add(Dense(units = 128,
                           activation = 'relu',
                           use_bias=True,
                           kernel_initializer='glorot_uniform',
                           bias_initializer='zeros'))
classificadorx2.add(Dropout(0.2))
classificadorx2.add(Dense(units = 128,
                          activation = 'relu'))
classificadorx2.add(Dropout(0.2))
classificadorx2.add(Dense(units = 1,
                          activation = 'sigmoid'))
optimizer = RMSprop(lr=0.001,
                    rho=0.9,
                    epsilon=1e-08,
                    decay=0.0)
classificadorx2.compile(optimizer = optimizer,
                        loss = 'binary_crossentropy',
                        metrics = ['accuracy'])
```

Fonte: Autor, 2022.

Na arquitetura do modelo, o mesmo conta com quatro camadas de convolução e suas camadas de filtro intermediárias, assim como quatro camadas densas, sendo a última o *perceptron* classificador. Ademais, as configurações permanecem exatamente iguais ao modelo X1. Lembrando que o modelo X1 em si mostrou-se, um tanto quanto inconsistente em sua configuração e resultados, sendo assim, o modelo X2 pode ser visto como uma tentativa de corrigir as inconsistências do modelo X1, tentando se chegar em um modelo ideal a partir do mesmo.

Figura 45 - Entrada das taxas de aprendizado classificador X2

```
50  epochs = 100
51
52  learning_rate = ReduceLROnPlateau(monitor='accuracy',
53                                    patience=3,
54                                    verbose=1,
55                                    factor=0.5,
56                                    min_lr=0.00001)
57
58  h_x2 = classificadorx2.fit(base_treino,
59                             steps_per_epoch = 100,
60                             epochs = epochs,
61                             validation_data = base_teste,
62                             callbacks = [learning_rate],
63                             verbose = 1,
64                             validation_steps = 25)
65
```

Fonte: Autor, 2022.

Todas as configurações referentes às épocas de processamento são mantidas iguais a do modelo X1.

Figura 46 - Taxas de aprendizado classificador X2 épocas iniciais

```
Epoch 1/100
100/100 [==============================] - 18s 165ms/step - loss: 0.4493 - accuracy: 0.8028 - val_loss: 0.8965 - val_accuracy: 0.5200
Epoch 2/100
100/100 [==============================] - 15s 147ms/step - loss: 0.3046 - accuracy: 0.8794 - val_loss: 0.6299 - val_accuracy: 0.5875
Epoch 3/100
100/100 [==============================] - 15s 147ms/step - loss: 0.2403 - accuracy: 0.9075 - val_loss: 0.4290 - val_accuracy: 0.7700
Epoch 4/100
100/100 [==============================] - 15s 147ms/step - loss: 0.2316 - accuracy: 0.9075 - val_loss: 0.1446 - val_accuracy: 0.9675
Epoch 5/100
100/100 [==============================] - 15s 147ms/step - loss: 0.2123 - accuracy: 0.9133 - val_loss: 0.4442 - val_accuracy: 0.8075
Epoch 6/100
100/100 [==============================] - 15s 150ms/step - loss: 0.2085 - accuracy: 0.9177 - val_loss: 1.4175 - val_accuracy: 0.8025
Epoch 7/100
100/100 [==============================] - 15s 149ms/step - loss: 0.1764 - accuracy: 0.9359 - val_loss: 2.7897 - val_accuracy: 0.8000
Epoch 8/100
100/100 [==============================] - 15s 149ms/step - loss: 0.1641 - accuracy: 0.9428 - val_loss: 4.2341 - val_accuracy: 0.7675
Epoch 9/100
100/100 [==============================] - 15s 146ms/step - loss: 0.1815 - accuracy: 0.9397 - val_loss: 1.8987 - val_accuracy: 0.8100
Epoch 10/100
100/100 [==============================] - 15s 149ms/step - loss: 0.1728 - accuracy: 0.9366 - val_loss: 0.5379 - val_accuracy: 0.8475
```

Fonte: Autor, 2022.

A acompanhando o início do processamento da CNN para cada modelo, pode-se observar, que a mesma rapidamente encontra seus primeiros padrões e retorna de seus testes valores crescentes de acurácia.

Figura 47 - Taxas de aprendizado classificador X2 épocas finais

```
Epoch 90/100
100/100 [==============================] - 15s 146ms/step - loss: 0.0862 - accuracy: 0.9623 - val_loss: 3.8950e-05 - val_accuracy: 1.0000
Epoch 91/100
100/100 [==============================] - 15s 147ms/step - loss: 0.0906 - accuracy: 0.9606 - val_loss: 4.0884e-05 - val_accuracy: 1.0000
Epoch 92/100
100/100 [==============================] - 15s 146ms/step - loss: 0.0808 - accuracy: 0.9717 - val_loss: 3.6750e-05 - val_accuracy: 1.0000
Epoch 93/100
100/100 [==============================] - 15s 147ms/step - loss: 0.0842 - accuracy: 0.9642 - val_loss: 2.4836e-05 - val_accuracy: 1.0000
Epoch 94/100
100/100 [==============================] - 15s 146ms/step - loss: 0.0868 - accuracy: 0.9642 - val_loss: 4.3916e-05 - val_accuracy: 1.0000
Epoch 95/100
100/100 [==============================] - 15s 148ms/step - loss: 0.0863 - accuracy: 0.9663 - val_loss: 2.1304e-05 - val_accuracy: 1.0000
Epoch 96/100
100/100 [==============================] - 15s 147ms/step - loss: 0.0894 - accuracy: 0.9686 - val_loss: 1.9424e-05 - val_accuracy: 1.0000
Epoch 97/100
100/100 [==============================] - 14s 144ms/step - loss: 0.0750 - accuracy: 0.9692 - val_loss: 1.5453e-05 - val_accuracy: 1.0000
Epoch 98/100
100/100 [==============================] - 14s 145ms/step - loss: 0.0883 - accuracy: 0.9673 - val_loss: 1.6755e-05 - val_accuracy: 1.0000
Epoch 99/100
100/100 [==============================] - 15s 146ms/step - loss: 0.0901 - accuracy: 0.9629 - val_loss: 2.9186e-05 - val_accuracy: 1.0000
Epoch 100/100
100/100 [==============================] - 15s 145ms/step - loss: 0.0757 - accuracy: 0.9667 - val_loss: 2.8362e-05 - val_accuracy: 1.0000
```

Fonte: Autor, 2022.

Ao encerrar o processamento, identificou-se que de acordo com suas últimas épocas de processamento, o modelo atual se mostrou ligeiramente mais eficiente que o anterior, do mesmo modo e em acordo com seus números. Tal modelo agora se tornou bastante consistente em seus valores de acurácia e perdas, tanto na fase de teste quanto de validação.

Figura 48 - Sumário do classificador X2

```
1  classificadorx2.summary()
2
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 128, 128, 128)     3584
_________________________________________________________________
batch_normalization_3 (Batch (None, 128, 128, 128)     512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 62, 62, 32)        36896
_________________________________________________________________
batch_normalization_4 (Batch (None, 62, 62, 32)        128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
batch_normalization_5 (Batch (None, 29, 29, 32)        128
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
batch_normalization_6 (Batch (None, 12, 12, 32)        128
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               147584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               16512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 129
=================================================================
Total params: 240,609
Trainable params: 240,161
Non-trainable params: 448
```

Fonte: Autor, 2022.

Analisando o sumário gerado para o modelo, observa-se que apesar de possuir mais camadas de processamento, rearranjou todos seus parâmetros, aproximadamente em torno de apenas 240 mil, considerado um número relativamente baixo se comparado à faixa dos milhões anteriormente gerados pelos demais modelos.

Essa característica, reflete que o modelo distribuiu de forma eficiente a base de dados para que fossem encontrados padrões pretendidos

Figura 49 - Aplicando o *Keras* para o classificador X1

```
keras2ascii(classificadorx2)

```

```
         OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

             Input   #####    128  128    3
            Conv2D    \|/  -------------------      3584     1.5%
              relu   #####    128  128  128
BatchNormalization    μ|σ  -------------------       512     0.2%
                     #####    128  128  128
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####     64   64  128
            Conv2D    \|/  -------------------     36896    15.3%
              relu   #####     62   62   32
BatchNormalization    μ|σ  -------------------       128     0.1%
                     #####     62   62   32
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####     31   31   32
            Conv2D    \|/  -------------------      9248     3.8%
              relu   #####     29   29   32
BatchNormalization    μ|σ  -------------------       128     0.1%
                     #####     29   29   32
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####     14   14   32
            Conv2D    \|/  -------------------      9248     3.8%
              relu   #####     12   12   32
BatchNormalization    μ|σ  -------------------       128     0.1%
                     #####     12   12   32
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####      6    6   32
           Flatten   ||||| -------------------         0     0.0%
                     #####        1152
             Dense   XXXXX -------------------    147584    61.3%
              relu   #####         128
           Dropout    | || -------------------         0     0.0%
                     #####         128
             Dense   XXXXX -------------------     16512     6.9%
              relu   #####         128
           Dropout    | || -------------------         0     0.0%
                     #####         128
             Dense   XXXXX -------------------     16512     6.9%
              relu   #####         128
           Dropout    | || -------------------         0     0.0%
                     #####         128
             Dense   XXXXX -------------------       129     0.1%
           sigmoid   #####           1
```

Fonte: Autor, 2022.

Analisando a distribuição por camada, vê-se grande parte das informações extraídas para os neurônios da rede foram concentradas nas camadas de convolução, fundamentais para que a partir das mesmas se encontrassem as principais características das imagens a serem replicadas para as camadas densas aprender seus padrões.

Figura 50 - Entrada para impressão da precisão e validação

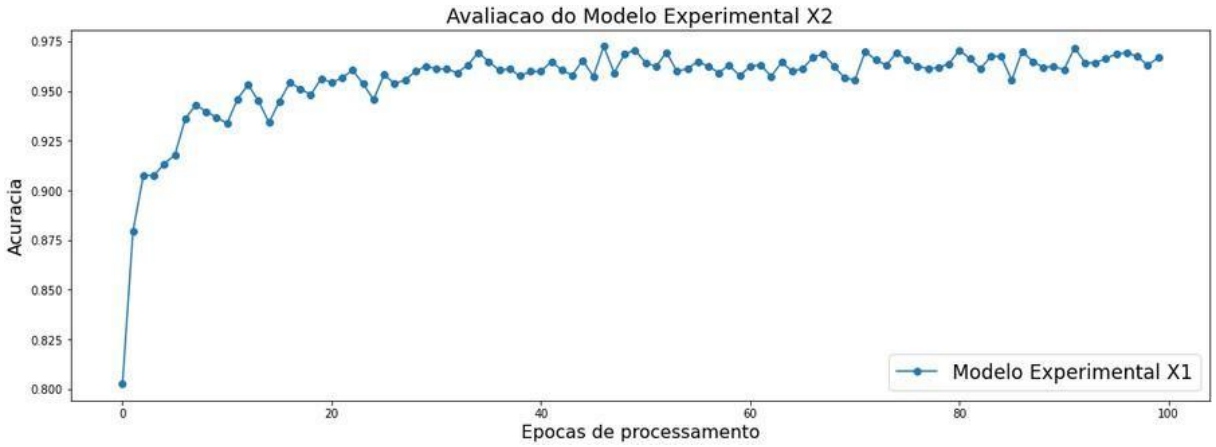
```
1  print('Precisão do treino: {0:.2f}%'.format(max(h_x2.history['accuracy']) * 100))
2  print('Precisão da validacao: {0:.2f}%'.format(max(h_x2.history['val_accuracy']) * 100))
3
```

```
Precisão do treino: 97.24%
Precisão da validacao: 100.00%
```

Fonte: Autor, 2022.

Exibindo em tela o resultado gerado pelas métricas internas de avaliação do modelo, nota-se que o mesmo retornou um valor de acurácia superior a 97%, dado que mesmo será considerado uma projetada margem de erro para mais ou para menos, é um dado/valor válido de acordo com os números retornados nos registros das épocas de processamento.

Gráfico 10 – Acurácia em função do número de épocas - classificador X2

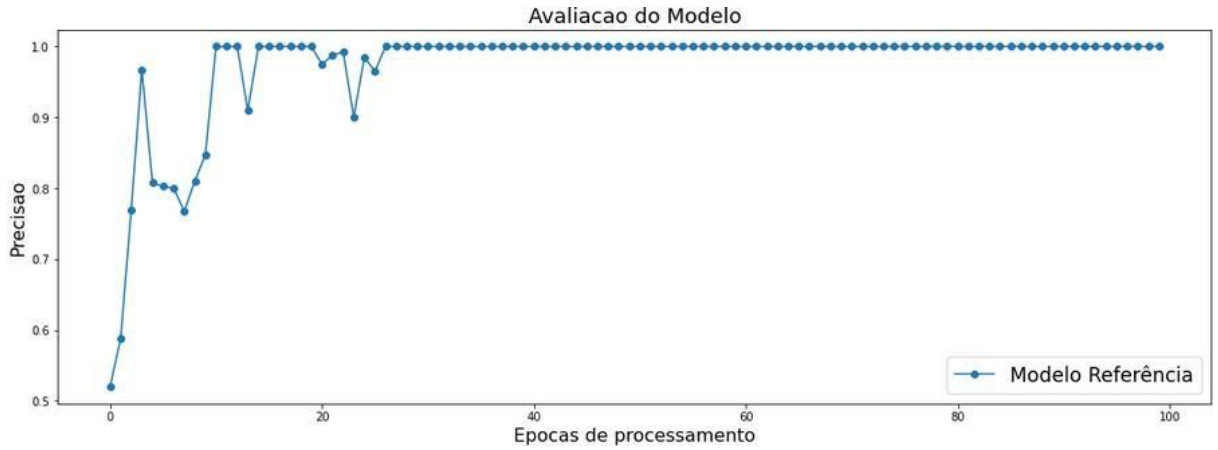


Fonte: Autor, 2022

No gráfico 10 observa-se a evolução da acurácia ao longo das épocas

de processamento, nota-se uma rápida ascensão logo nas primeiras épocas e a relativa constância mantida ao longo de todo o resto do processamento.

Gráfico 11 - Precisão em função do número de épocas - classificador X2



Fonte: Autor, 2022.

Analisando o gráfico 11, referente a precisão do modelo, pode-se confirmar que o mesmo logo após a vigésima sétima época de processamento já conseguiu encontrar e manter uma margem de precisão muito alta.

Gráfico 12 - Aprendizado em função do número de épocas - classificador X2

Fonte: Autor, 2022

Ao finalizar a interpretação desse modelo, foi possível notar no que se refere a taxa de aprendizado que o modelo antes mesmo do processamento da metade das épocas definidas, já havia encontrado um valor ótimo, replicado ao

longo de todo o restante das épocas de processamento.

4.13 Classificador X3

Uma vez que apresentado o modelo X2 com excelente performance e eficiência, oferecendo grandes melhorias quando comparado aos modelos anteriores.

Destaca-se a possibilidade de gerar-se um último modelo experimental denominado X3, o qual fez a utilização de diferentes ferramentas, configurações e aprimoramentos, buscando encontrar um modelo tão consistente quanto ou ainda melhor que o modelo X2.

Para tanto, seguiu-se os mesmos passos de implementação da CNN, podendo ser visualizados a seguir na figura 50.

Figura 51 - Entrada de um novo código de transposição de matrizes

```
classificadorx3 = Sequential()
classificadorx3.add(Conv2D(128,
                           kernel_size = (3,3),
                           padding = 'same',
                           input_shape = (128,128,3),
                           activation = 'relu'))
classificadorx3.add(BatchNormalization())
classificadorx3.add(MaxPooling2D(pool_size = (2,2)))
classificadorx3.add(Conv2D(128,
                           (3,3),
                           activation = 'relu'))
classificadorx3.add(BatchNormalization())
classificadorx3.add(MaxPooling2D(pool_size = (2,2)))
classificadorx3.add(Conv2D(128,
                           (3,3),
                           activation = 'relu'))
classificadorx3.add(Conv2DTranspose(128,
                                    kernel_size = (3,3),
                                    strides = (1, 1),
                                    padding = 'valid',
                                    output_padding = None,
                                    data_format = None,
                                    dilation_rate = (1, 1),
                                    activation = None,
                                    use_bias = True,
                                    kernel_initializer = 'glorot_uniform'))
classificadorx3.add(Conv2D(128,
                           (3,3),
                           activation = 'relu'))
classificadorx3.add(BatchNormalization())
classificadorx3.add(MaxPooling2D(pool_size = (2,2)))
classificadorx3.add(Conv2DTranspose(128,
                                    kernel_size = (3,3),
                                    strides = (1, 1),
                                    padding = 'valid',
                                    output_padding = None,
                                    data_format = None,
                                    dilation_rate = (1, 1),
                                    activation = None,
                                    use_bias = True,
                                    kernel_initializer = 'glorot_uniform'))
```

Fonte: Autor, 2022.

É possível afirmar que o código é o grande diferencial deste modelo em comparação ao modelo anterior se dá pelo emprego de duas camadas de

transposição de matrizes. Tal técnica realiza uma varredura por blocos ou em determinadas situações na imagem por completo, transformando dentro de tal bloco linhas em colunas e vice-versa, mantendo seus valores originais, apenas realocando pixels dentro do bloco.

A técnica de transposição de matrizes é muito útil para casos, os quais se pretende que os dados extraídos de uma imagem para uma matriz sejam mapeados, conforme a posição original da imagem, abrindo um leque de possibilidades para que características importantes da imagem fonte, sejam identificadas independentemente da orientação da imagem, assim como de suas características morfológicas.

A seguir na figura 51 realizou-se a entrada de um novo código mais aprimorado no sentido aprimorar o classificador.

Figura 52 - Entrada de um novo código aprimorado

```
42  classificadorx3.add(Conv2D(128,
43                             (3,3),
44                             activation = 'relu'))
45  classificadorx3.add(BatchNormalization())
46  classificadorx3.add(MaxPooling2D(pool_size = (2,2)))
47  classificadorx3.add(Flatten())
48  classificadorx3.add(Dense(units = 128,
49                             activation = 'relu',
50                             use_bias=True,
51                             kernel_initializer='glorot_uniform',
52                             bias_initializer='zeros'))
53  classificadorx3.add(Dropout(0.1))
54  classificadorx3.add(Dense(units = 256,
55                             activation = 'relu',
56                             use_bias=True,
57                             kernel_initializer='glorot_uniform',
58                             bias_initializer='zeros'))
59  classificadorx3.add(Dropout(0.1))
60  classificadorx3.add(Dense(units = 256,
61                             activation = 'relu',
62                             use_bias=True,
63                             kernel_initializer='glorot_uniform',
64                             bias_initializer='zeros'))
65  classificadorx3.add(Dropout(0.1))
66  classificadorx3.add(Dense(units = 128,
67                             activation = 'relu',
68                             use_bias=True,
69                             kernel_initializer='glorot_uniform',
70                             bias_initializer='zeros'))
71  classificadorx3.add(Dropout(0.1))
72  classificadorx3.add(Dense(units = 1,
73                            activation = 'sigmoid'))
74  optimizer = Adam(learning_rate=0.001,
75                   beta_1=0.9,
76                   beta_2=0.999,
77                   epsilon=1e-07,
78                   amsgrad=False,
79                   name="Adam")
80  classificadorx3.compile(optimizer = optimizer,
81                          loss = 'binary_crossentropy',
82                          metrics = ['accuracy'])
83
```

Fonte: Autor, 2022.

Seguindo com a estrutura do modelo retratada na figura 51, para as camadas densas são definidos manualmente alguns parâmetros adicionais, realizando assim, alguns ajustes finos, como por exemplo o uso obrigatório de camadas de (*bias)* que significa uma entrada de valor "1" associada a um peso "b" em cada neurônio.

As camadas intermediárias são utilizadas para ajuste fino dos pesos encontrados entre neurônios, mas também permitem um eficiente mecanismo de retroalimentação da rede.

Modelos comuns de redes neurais artificiais normalmente são construídos de modo sequencial onde as camadas processam dados somente em um sentido, desde a origem até o classificador.

Todavia, é perfeitamente possível configurar o modelo de modo que o mesmo se retroalimente, replicando padrões encontrados, tanto para camadas subsequentes quanto para camadas anteriores, aumentando consideravelmente a margem de acertos dos pesos encontrados entre neurônios e entre camadas. Para isso, dá-se a entrada como visto na figura 52.

Figura 53 - Entrada das taxas de aprendizado classificador X2

```
84  epochs = 100
85
86  learning_rate = ReduceLROnPlateau(monitor='accuracy',
87                                    patience=2,
88                                    verbose=1,
89                                    factor=0.5,
90                                    min_lr=0.0001)
91
92  h_x3 = classificadorx3.fit(base_treino,
93                             steps_per_epoch = 100,
94                             epochs = epochs,
95                             validation_data = base_teste,
96                             callbacks = [learning_rate],
97                             verbose = 1,
98                             validation_steps = 25)
99
```

Fonte: Autor, 2022.

As demais configurações se mantêm sem alteração, tais como os modelos de classificação anteriores.

Todavia, para fins de validação da acurácia introduz-se as taxas de aprendizado conforme figura 53, onde pode-se observar as épocas iniciais.

Figura 54 - Taxas de aprendizado classificador X2 épocas iniciais

```
Epoch 1/100
100/100 [==============================] - 16s 152ms/step - loss: 0.4968 - accuracy: 0.7831 - val_loss: 0.9320 - val_accuracy: 0.4750
Epoch 2/100
100/100 [==============================] - 15s 150ms/step - loss: 0.3441 - accuracy: 0.8662 - val_loss: 2.2847 - val_accuracy: 0.5000
Epoch 3/100
100/100 [==============================] - 15s 147ms/step - loss: 0.2743 - accuracy: 0.8913 - val_loss: 0.6848 - val_accuracy: 0.5150
Epoch 4/100
100/100 [==============================] - 15s 148ms/step - loss: 0.2552 - accuracy: 0.9089 - val_loss: 0.2421 - val_accuracy: 0.8550
Epoch 5/100
100/100 [==============================] - 15s 147ms/step - loss: 0.2260 - accuracy: 0.9102 - val_loss: 1.0073 - val_accuracy: 0.7900
Epoch 6/100
100/100 [==============================] - 15s 148ms/step - loss: 0.1991 - accuracy: 0.9253 - val_loss: 0.2231 - val_accuracy: 0.8900
Epoch 7/100
100/100 [==============================] - 15s 148ms/step - loss: 0.1826 - accuracy: 0.9231 - val_loss: 0.7132 - val_accuracy: 0.8325
Epoch 8/100
100/100 [==============================] - 15s 147ms/step - loss: 0.1829 - accuracy: 0.9315 - val_loss: 2.1319 - val_accuracy: 0.7400
Epoch 9/100
100/100 [==============================] - 15s 151ms/step - loss: 0.1566 - accuracy: 0.9391 - val_loss: 0.0418 - val_accuracy: 0.9875
Epoch 10/100
100/100 [==============================] - 15s 150ms/step - loss: 0.1806 - accuracy: 0.9278 - val_loss: 2.1435 - val_accuracy: 0.7925
```

Fonte: Autor, 2022.

Da mesma forma, como realizado para os modelos anteriores, uma vez colocada em execução a RNA, pode-se acompanhar o *log* de suas épocas de processamento. Do mesmo modo como observado nos exemplos anteriores, o modelo começa com uma margem de acurácia baixa, rapidamente aumentada à medida que os padrões corretos começam a serem encontrados, como observado na figura 54.

Figura 55 - Taxas de aprendizado classificador X2 épocas finais

```
Epoch 90/100
100/100 [==============================] - 15s 152ms/step - loss: 0.0483 - accuracy: 0.9837 - val_loss: 0.2477 - val_accuracy: 0.8975
Epoch 91/100
100/100 [==============================] - 15s 149ms/step - loss: 0.0562 - accuracy: 0.9762 - val_loss: 1.1672 - val_accuracy: 0.8050
Epoch 92/100
100/100 [==============================] - 15s 149ms/step - loss: 0.0469 - accuracy: 0.9774 - val_loss: 0.0816 - val_accuracy: 0.9575
Epoch 93/100
100/100 [==============================] - 15s 148ms/step - loss: 0.0508 - accuracy: 0.9818 - val_loss: 0.0181 - val_accuracy: 0.9900
Epoch 94/100
100/100 [==============================] - 15s 151ms/step - loss: 0.0429 - accuracy: 0.9824 - val_loss: 0.1380 - val_accuracy: 0.9200
Epoch 95/100
100/100 [==============================] - 15s 150ms/step - loss: 0.0412 - accuracy: 0.9818 - val_loss: 1.1212e-04 - val_accuracy: 1.0000
Epoch 96/100
100/100 [==============================] - 15s 149ms/step - loss: 0.0542 - accuracy: 0.9768 - val_loss: 0.2769 - val_accuracy: 0.8900
Epoch 97/100
100/100 [==============================] - 15s 152ms/step - loss: 0.0576 - accuracy: 0.9731 - val_loss: 0.1481 - val_accuracy: 0.9225
Epoch 98/100
100/100 [==============================] - 15s 147ms/step - loss: 0.0521 - accuracy: 0.9799 - val_loss: 0.1279 - val_accuracy: 0.9375
Epoch 99/100
100/100 [==============================] - 15s 150ms/step - loss: 0.0501 - accuracy: 0.9818 - val_loss: 0.2432 - val_accuracy: 0.8925
Epoch 100/100
100/100 [==============================] - 15s 146ms/step - loss: 0.0447 - accuracy: 0.9812 - val_loss: 0.0673 - val_accuracy: 0.9775
```

Fonte: Autor, 2022.

Por fim, terminado o processamento de todas as épocas, o registro nos mostra uma alta acurácia, em torno de 98%, que acompanhada de uma taxa de perda muito baixa, em torno de 5% nos confirma a alta eficiência do modelo.

Figura 56 - Sumário do classificador X3

```
1  classificadorx3.summary()
2
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 128, 128, 128)     3584
_________________________________________________________________
batch_normalization_7 (Batch (None, 128, 128, 128)     512
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 62, 62, 128)       147584
_________________________________________________________________
batch_normalization_8 (Batch (None, 62, 62, 128)       512
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 31, 31, 128)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 29, 29, 128)       147584
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 31, 31, 128)       147584
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 29, 29, 128)       147584
_________________________________________________________________
batch_normalization_9 (Batch (None, 29, 29, 128)       512
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 16, 16, 128)       147584
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 128)       147584
_________________________________________________________________
batch_normalization_10 (Batc (None, 14, 14, 128)       512
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               802944
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 256)               33024
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_14 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
```

Fonte: Autor, 2022.

De acordo com o sumário do classificador X3, foram obtidos cerca de um milhão e oitocentas mil parâmetros, com a adição de camadas de erros (*bias*) intermediárias, como também, camadas de transposição de matrizes com suas respectivas camadas intermediárias.

Na sequência dá-se a aplicação do *Keras* para o classificador X3.

Figura 57 - Aplicando o *Keras* para o classificador X3

```
1  keras2ascii(classificadorx3)
2
```

```
         OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

             Input   #####    128  128    3
            Conv2D    \|/  -------------------      3584     0.2%
              relu   #####    128  128  128
BatchNormalization    μ|σ  -------------------       512     0.0%
                     #####    128  128  128
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####     64   64  128
            Conv2D    \|/  -------------------    147584     8.1%
              relu   #####     62   62  128
BatchNormalization    μ|σ  -------------------       512     0.0%
                     #####     62   62  128
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####     31   31  128
            Conv2D    \|/  -------------------    147584     8.1%
              relu   #####     29   29  128
   Conv2DTranspose    /|\  -------------------    147584     8.1%
                     #####     31   31  128
            Conv2D    \|/  -------------------    147584     8.1%
              relu   #####     29   29  128
BatchNormalization    μ|σ  -------------------       512     0.0%
                     #####     29   29  128
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####     14   14  128
   Conv2DTranspose    /|\  -------------------    147584     8.1%
                     #####     16   16  128
            Conv2D    \|/  -------------------    147584     8.1%
              relu   #####     14   14  128
BatchNormalization    μ|σ  -------------------       512     0.0%
                     #####     14   14  128
      MaxPooling2D   Y max -------------------         0     0.0%
                     #####      7    7  128
           Flatten   |||||  -------------------         0     0.0%
                     #####         6272
             Dense   XXXXX  -------------------    802944    44.0%
              relu   #####          128
           Dropout    | ||  -------------------         0     0.0%
                     #####          128
             Dense   XXXXX  -------------------     33024     1.8%
              relu   #####          256
           Dropout    | ||  -------------------         0     0.0%
                     #####          256
             Dense   XXXXX  -------------------     65792     3.6%
              relu   #####          256
           Dropout    | ||  -------------------         0     0.0%
                     #####          256
             Dense   XXXXX  -------------------     32896     1.8%
              relu   #####          128
           Dropout    | ||  -------------------         0     0.0%
                     #####          128
```

Fonte: Autor, 2022

Analisando a distribuição por camada destaca-se, finalmente um modelo que mesmo baseado em um grande número de parâmetros, distribuiu e rearranjou suas estruturas de modo que cada tipo de camada da rede neural recebeu valores adequados de amostras, refletindo na melhoria de padrões encontrados.

Figura 58 - Entrada para impressão da precisão e validação

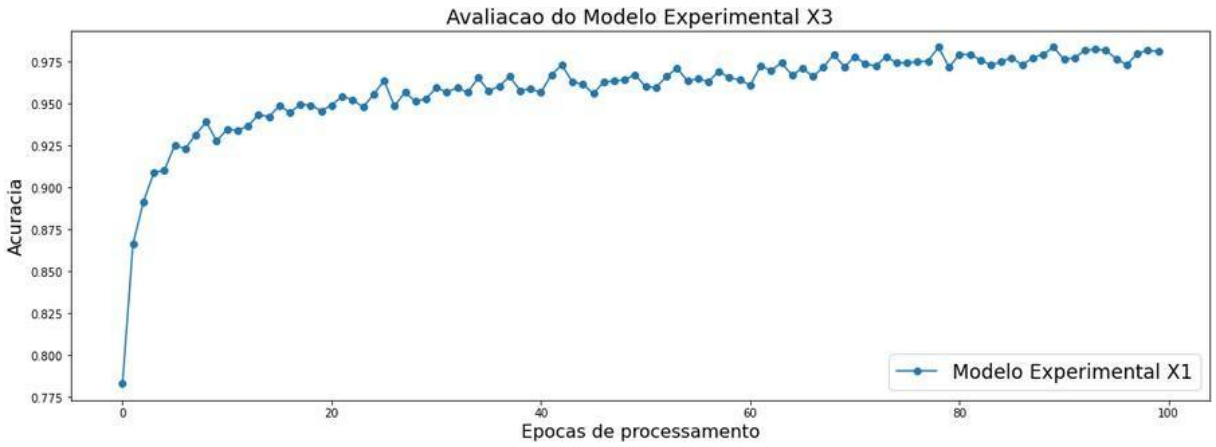
```
1  print('Precisão do treino: {0:.2f}%'.format(max(h_x3.history['accuracy']) * 100))
2  print('Precisão da validacao: {0:.2f}%'.format(max(h_x3.history['val_accuracy']) * 100))
3
```

```
Precisão do treino: 98.37%
Precisão da validacao: 100.00%
```

Fonte: Autor, 2022.

Exibindo em tela da figura 57 o retorno padrão gerado pelo modelo, o classificador apresentou uma margem de acertos em seus testes superior a 98%.

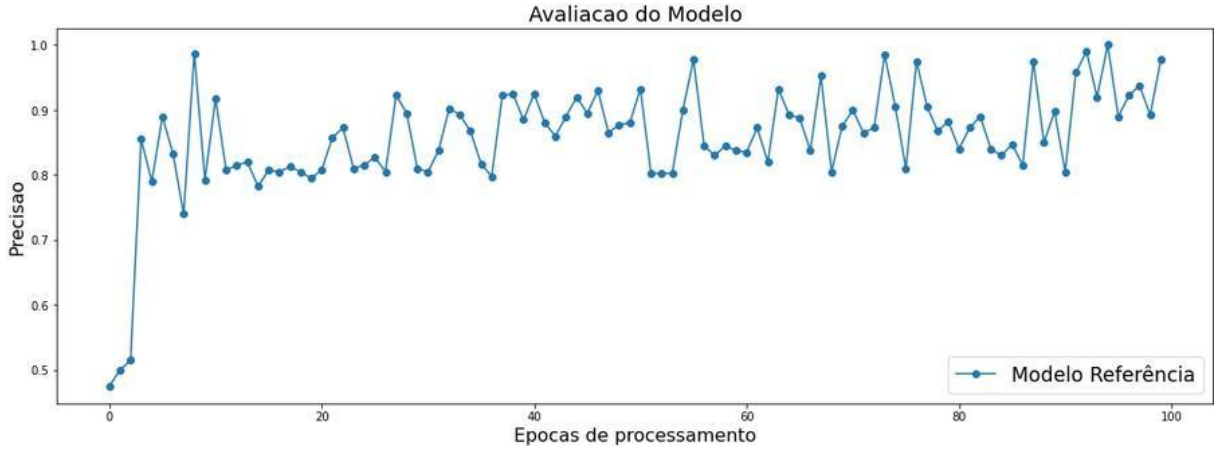
Gráfico 13 - Acurácia em função do número de épocas - classificador X3



Fonte: Autor, 2022.

Analisando gráfico 13 e sua evolução do modelo no que diz respeito a acurácia, visualizou-se uma gradativa melhoria, sendo crescente a partir da vigésima época de processamento.

Gráfico 14 - Precisão em função do número de épocas - classificador X3



Fonte: Autor, 2022.

Observando o gráfico 14 referente a precisão do modelo, nota-se uma certa inconsistência, muito provavelmente ocasionada por *overfitting* do modelo em função de seu grande número de parâmetros.

Gráfico 15 - Aprendizado em função do número de épocas - classificador X3

Fonte: Autor, 2022.

Ao finalizar o classificador X3, acompanhando o histórico referente a taxa de aprendizado, conforme gráfico 15, a mesma já a partir da época 30 de processamento do modelo, encontra-se um melhor valor estimado para aprendizado de máquina, replicando o mesmo até o final das épocas de processamento.

5 RESULTADOS E DISCUSSÃO

Entre os **resultados** encontrados, estão os produzidos pelos modelos que foram treinados, testados, validados por métricas de avaliação de Aprendizado de Máquina a *Machine Learning*.

Sendo que cada modelo teve sua arquitetura, incrementada com camadas de extração de características e de processamento das imagens.

Durante os processos de validação optou-se pela acurácia e pela taxa de perdas que é inversamente proporcional a uma à outra.

A figura 58 mostra os 5 modelos classificados, segundo a acurácia e validação da acurácia.

Figura 59 - Precisão da CNN e sua validação

```
print('MODELO BASE')
print('Precisão do treino: {0:.2f}%'.format(max(h_0.history['accuracy']) * 100))
print('Precisão da validacao: {0:.0f}%'.format(max(h_0.history['val_accuracy']) * 100))
print('_____________________________________________________________________')
print('CLASSIFICADOR REFERÊNCIA')
print('Precisão do treino: {0:.2f}%'.format(max(h_1.history['accuracy']) * 100))
print('Precisão da validacao: {0:.0f}%'.format(max(h_1.history['val_accuracy']) * 100))
print('_____________________________________________________________________')
print('CLASSIFICADOR X1')
print('Precisão do treino: {0:.2f}%'.format(max(h_x1.history['accuracy']) * 100))
print('Precisão da validacao: {0:.0f}%'.format(max(h_x1.history['val_accuracy']) * 100))
print('_____________________________________________________________________')
print('CLASSIFICADOR X2')
print('Precisão do treino: {0:.2f}%'.format(max(h_x2.history['accuracy']) * 100))
print('Precisão da validacao: {0:.0f}%'.format(max(h_x2.history['val_accuracy']) * 100))
print('_____________________________________________________________________')
print('CLASSIFICADOR X3')
print('Precisão do treino: {0:.2f}%'.format(max(h_x3.history['accuracy']) * 100))
print('Precisão da validacao: {0:.0f}%'.format(max(h_x3.history['val_accuracy']) * 100))
print('_____________________________________________________________________')

```

```
MODELO BASE
Precisão do treino: 98.00%
Precisão da validacao: 92%
_____________________________________________________________________
CLASSIFICADOR REFERÊNCIA
Precisão do treino: 95.85%
Precisão da validacao: 100%
_____________________________________________________________________
CLASSIFICADOR X1
Precisão do treino: 93.22%
Precisão da validacao: 100%
_____________________________________________________________________
CLASSIFICADOR X2
Precisão do treino: 97.24%
Precisão da validacao: 100%
_____________________________________________________________________
CLASSIFICADOR X3
Precisão do treino: 98.37%
Precisão da validacao: 100%
_____________________________________________________________________
```

Fonte: Autor, 2022.

Desconsiderando os modelos Base e Referência, dos modelos criados exclusivamente para a base de dados, todos os modelos apresentaram ótimos resultados dentro de suas limitações e propósitos.

Em termos de aplicação prática, pode-se considerar que qualquer um dos modelos, possuem plenas condições de classificar corretamente imagens com uma altíssima margem de acertos, cabendo apenas ao desenvolvedor optar pelo modelo que julgue mais consistente e/ou de menor margem de erro de acordo com suas particularidades.

5.1 Testes Aplicados em Amostras

Assim que os modelos são treinados, dá-se a necessidade de carregar suas configurações de aprendizado de máquina e realizar testes reais processando novas amostras nos mesmos.

5.2 Classificador de Referência

Para realizar os testes, não foi necessário treinar o modelo para cada teste e sim carregar suas configurações de aprendizado de máquina para a CNN, então, foi utilizado essas referências para realizar as devidas classificações.

A partir do código, foi necessário realizar o carregamento do tipo de modelo e suas configurações.

Figura 60 - Entrada para carregar imagem normal (referência)

```
img_teste = load_img('NORMAL(2398).jpg',
                     target_size = (128, 128))

img_plot = PIL.Image.open('NORMAL(2398).jpg')

plt.figure(figsize=(8,8))
plt.imshow(img_plot)
plt.show()

img_teste = image.img_to_array(img_teste)
img_teste /= 255
img_teste = np.expand_dims(img_teste,
                           axis = 0)

resultado_teste = model.predict(img_teste)

resultado_final = resultado_teste

```

Fonte: Autor, 2022.

Em seguida foi criada uma estrutura de código para realizar o carregamento da imagem/amostra a ser processada, neste processo foram realizadas algumas simples validações, entre elas, a principal foi converter a imagem para uma matriz de pixels com a finalidade ser lida e interpretada pela CNN.

Como exemplo a figura 60 mostra o RX de uma coluna lombar sem escoliose que no algoritmo será utilizada para que os casos com escoliose sejam identificados pelo algoritmo.

Figura 61 - Imagem de RX Normal

Fonte: Autor, 2022

Figura 62 - Código dos resultados normal (referência)

```
1  if resultado_final[0] < 0.5:
2      print('Radiografia de Coluna Lombar com aspecto NORMAL.')
3  else:
4      print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')
5
```

```
Radiografia de Coluna Lombar com aspecto NORMAL.
```

Fonte: Autor, 2022.

Figura 63 - Código dos resultados normal (referência)

```
1  if resultado_final[0] < 0.5:
2      print('Radiografia de Coluna Lombar com aspecto NORMAL.')
3  else:
4      print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')
5
```

```
Radiografia de Coluna Lombar com aspecto NORMAL.
```

Fonte: Autor, 2022.

Ao executar o bloco de código visto na figura 61 o RX foi devidamente processado pela rede neural recebendo um valor de referência, que neste caso ficou atribuído a variável “resultado final”.

Com base nesta variável e seu respectivo valor, pôde-se criar uma simples estrutura condicional que lê o valor de tal variável, verificando se o mesmo for menor que 0.5, classificando a amostra como NORMAL, assim como se tal valor for superior a 0.5, tal amostra é classificada como ESCOLIOSE.

Neste exemplo, de acordo com as características interpretadas na imagem, a mesma foi classificada como uma radiografia de coluna lombar de aspecto normal.

Figura 64 - Entrada para carregar imagem patológica (referência)

```
img_teste = load_img('PATOLOGICA_0_8131.jpeg',
                     target_size = (128, 128))

img_plot = PIL.Image.open('PATOLOGICA_0_8131.jpeg')

plt.figure(figsize=(8,8))
plt.imshow(img_plot)
plt.show()

img_teste = image.img_to_array(img_teste)
img_teste /= 255
img_teste = np.expand_dims(img_teste,
                           axis = 0)

resultado_teste = model.predict(img_teste)

resultado_final = resultado_teste

```

Fonte: Autor, 2022.

O código da figura 62, representa a maneira de inserir uma imagem para teste, esta por sua vez é um exemplo de imagem de RX patológico, vide figura 67, sendo submetida ao mesmo teste que o RX normal.

Figura 65 - Imagem de RX Patológica (referência)

Fonte: Autor, 2022

Figura 66 - Código dos resultados escoliose (referência)

```
1  if resultado_final[0] < 0.5:
2      print('Radiografia de Coluna Lombar com aspecto NORMAL.')
3  else:
4      print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')
5
```

```
Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.
```

Fonte: Autor, 2022.

Desta forma, a programação da figura 64, traz o resultado retornando que de acordo com suas características tal do RX, o qual neste caso representa uma radiografia de coluna compatível para escoliose, ou seja, patológica.

5.3 Classificador Modelo X1

A sequência de impressões, conforme figura 65 e 66 a seguir, representam os códigos aplicados ao modelo X1.

Figura 67 - Entrada do classificador modelo (X1)

```
model = Sequential()
model = load_model('classificador_x1_model.h5')

```

Fonte: Autor, 2022.

Figura 68 - Entrada para carregar imagem normal (X1)

```
img_teste = load_img('NORMAL(2398).jpg',
                     target_size = (128, 128))

img_plot = PIL.Image.open('NORMAL(2398).jpg')

resultado_teste = model.predict(img_teste)

resultado_final = resultado_teste

```

Fonte: Autor, 2022.

A figura 71 traz um outro exemplo de imagem do banco de dados considerada um RX normal.

Figura 69 - Imagem de RX Normal (X1)

Fonte: Autor, 2022.

Para fins didáticos e demonstrativos da construção do código utilizado as figuras 67 e 68 trazem a designação de retorno com o resultado da classificação.

Figura 70 - Código dos resultados normal (X1)

```
if resultado_final[0] < 0.5:
    print('Radiografia de Coluna Lombar com aspecto NORMAL.')
else:
    print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')

```

```
Radiografia de Coluna Lombar com aspecto NORMAL.
```

Fonte: Autor, 2022.

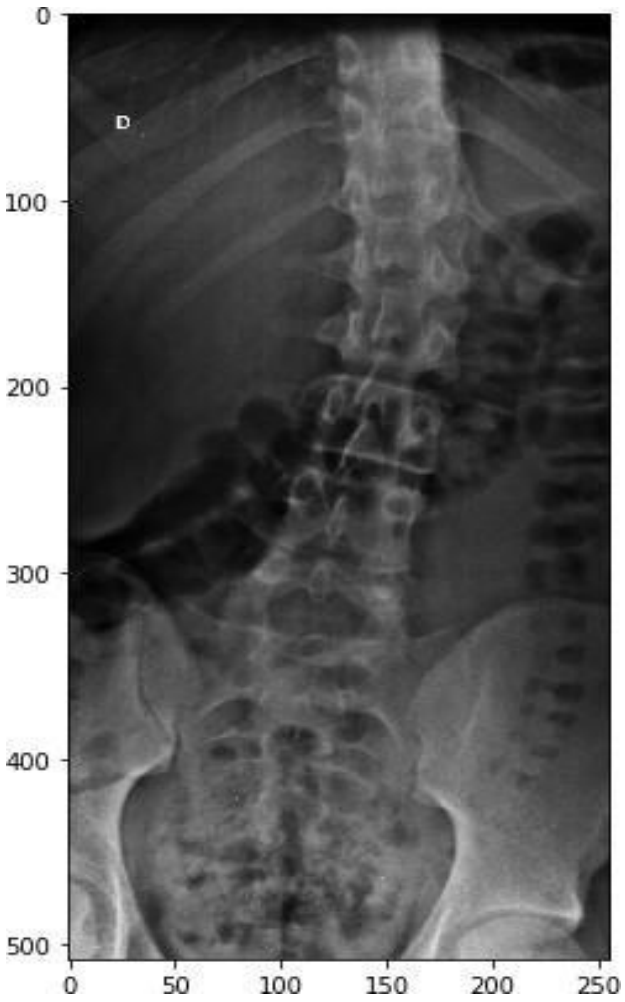
Figura 71 - Entrada para carregar imagem patológica (X1)

```
img_teste = load_img('PATOLOGICA(92943).jpg',
                     target_size = (128, 128))

img_plot = PIL.Image.open('PATOLOGICA(92943).jpg')

resultado_teste = model.predict(img_teste)

resultado_final = resultado_teste

```

Fonte: Autor, 2022.

Da mesma forma que foi representada no modelo anterior, neste momento insere-se na CNN, vide figura 70, uma imagem de RX considerada patológica.

Figura 72 - Imagem de RX Patológica (X1)



Fonte: Autor, 2022.

Figura 73 - Código dos resultados escoliose (X1)

```
1  if resultado_final[0] < 0.5:
2      print('Radiografia de Coluna Lombar com aspecto NORMAL.')
3  else:
4      print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')
5
```

```
Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.
```

Fonte: Autor, 2022.

5.4 Classificador Modelo X2

A cada novo modelo criado, neste caso o X2, novamente faz-se a programação como visto nas figuras 72 e 73, a fim de obter os dados pretendidos.

Figura 74 - Entrada do classificador modelo (X2)

```
model = Sequential()
model = load_model('/classificador_x2_model.h5')

```

Fonte: Autor, 2022.

Figura 75 - Entrada para carregar imagem normal (X2)

```
img_teste = load_img('NORMAL_0_7911.jpeg',
                     target_size = (128, 128))

img_plot = PIL.Image.open('NORMAL_0_7911.jpeg')

resultado_teste = model.predict(img_teste)

resultado_final = resultado_teste

```

Fonte: Autor, 2022

Neste momento retratado na figura 74, evidencia-se, mas um exemplo de imagem de RX normal.

Figura 76 - Imagem de RX Normal (X2)

Fonte: Autor, 2022.

Da mesma maneira repete-se os procedimentos e códigos de programação para a imagem de RX com escoliose.

5.5 Classificador Modelo X3

Os códigos representados nas figuras 75 e 77, confirmam por meio da programação a inserção de mais uma imagem para teste do modelo.

Figura 77 - Entrada do classificador modelo (X3)

```
model = Sequential()
model = load_model('classificador_x3_model.h5')

```

Fonte: Autor, 2022.

Figura 78 - Entrada para carregar imagem normal (X3)

```
img_teste = load_img('PATOLOGICA_0_8131.jpeg',
                     target_size = (128, 128))

img_plot = PIL.Image.open('PATOLOGICA_0_8131.jpeg')

resultado_teste = model.predict(img_teste)

resultado_final = resultado_teste

```

Fonte: Autor, 2022.

E finalmente nesta terceira modelagem, novas imagens de RX, são observadas e testadas para fins de validação do modelo, conforme o exemplo a seguir com características patológicas presentes na figura 77

Figura 79 - Imagem de RX Patológica (X3)

Fonte: Autor, 2022.

Sendo elucidado no código da figura 78 uma identificação positiva para coluna lombar com aspecto de escoliose.

Figura 80 - Código dos resultados escoliose (X3)

```
if resultado_final[0] < 0.5:
    print('Radiografia de Coluna Lombar com aspecto NORMAL.')
else:
    print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')

```

```
Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.
```

Fonte: Autor, 2022.

5.6 Comparativo entre todos os modelos

Depois dos modelos treinados e testados, considerou-se necessário realizar comparações entre os mesmos para determinar qual é o mais adequado ao propósito deste estudo de classificação a partir de imagens de raio X, como destacado no código da figura 79.

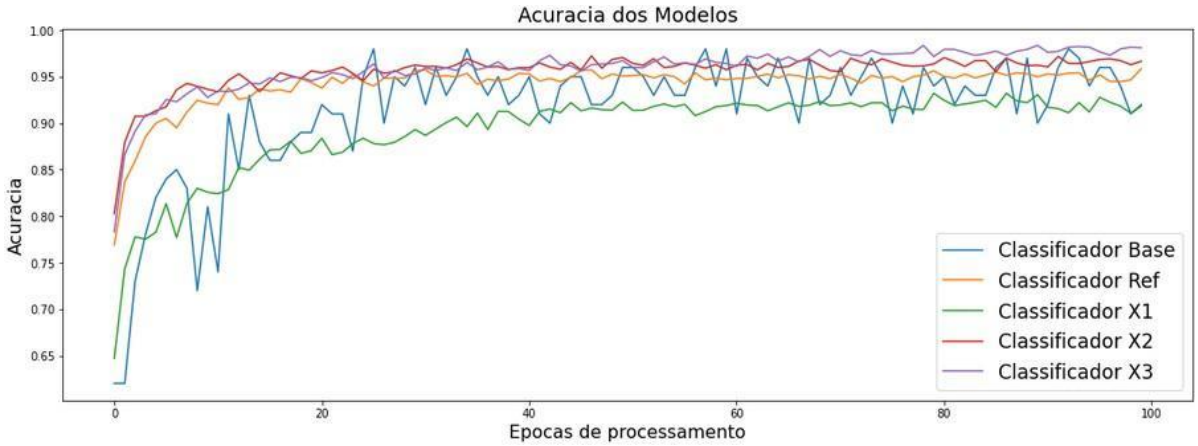
Figura 81 - Comparações entre os classificadores

```
plt.rcParams['figure.figsize'] = (18.0, 6.0)
plt.plot(h_0.history['accuracy'])
plt.plot(h_1.history['accuracy'])
plt.plot(h_x1.history['accuracy'])
plt.plot(h_x2.history['accuracy'])
plt.plot(h_x3.history['accuracy'])
plt.legend(['Classificador Base',
            'Classificador Ref',
            'Classificador X1',
            'Classificador X2',
            'Classificador X3'],
           loc = 'lower right', fontsize = 'xx-large')
plt.xlabel('Epocas de processamento', fontsize=16)
plt.ylabel('Acuracia', fontsize=16)
plt.title('Acuracia dos Modelos', fontsize=18)
plt.show()

```

Fonte: Autor, 2022.

Em relação à acurácia, fora uma representação de todos os cenários em diferentes épocas, conforme mostra o gráfico 16.

Gráfico 16 - Acurácia dos classificadores



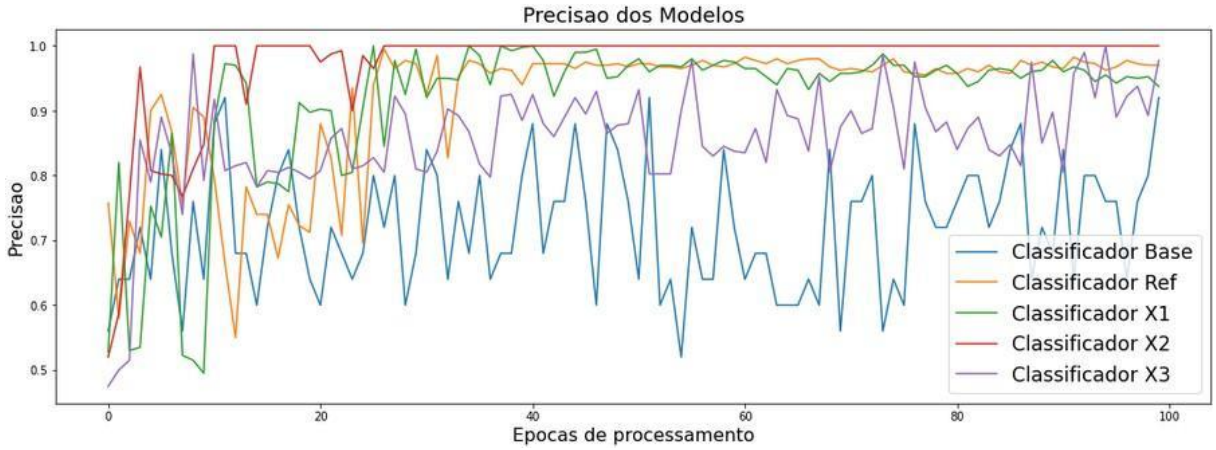
Fonte: Autor, 2022.

Ao se tratar da acurácia dos modelos, pôde-se observar que mesmos os modelos base e referência apresentaram resultados superiores a 85%, porém, como foi evidenciado anteriormente, de acordo com algumas inconsistências encontradas nos modelos, tais inconsistências são números que podem refletir alguns falsos positivos.

De acordo com a projeção de suas linhas, note-se que de acurácia dos modelos X2 e X3 foram os que se mostraram melhores.

No tocante a precisão dos modelos, de acordo com o gráfico 17, observa-se que no treinamento e no teste de cada um, houve algumas inconsistências a serem levadas em consideração, não no sentido de descartá-las e sim avaliar seus valores de margens de acertos.

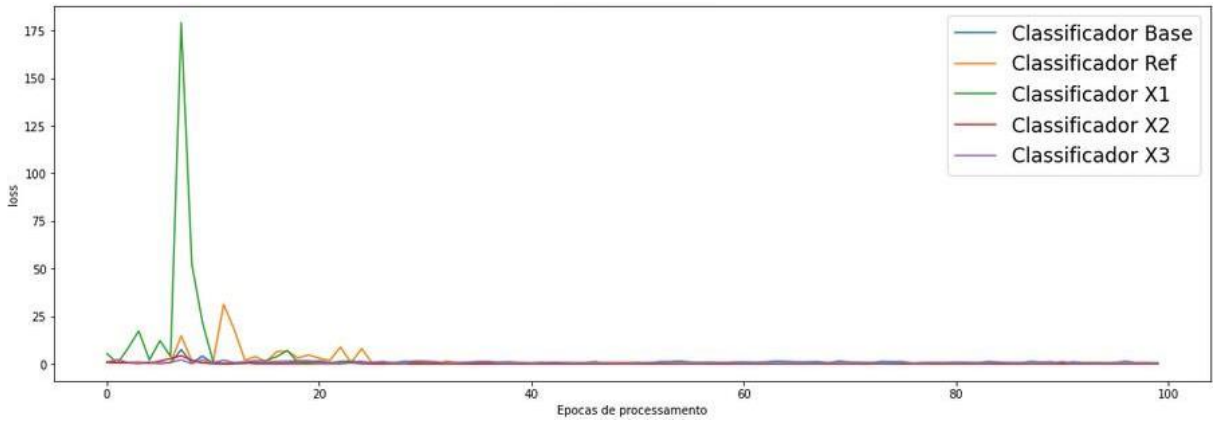
Gráfico 17 - Precisão dos classificadores



Fonte: Autor, 2022.

Considerando estes modelos comparados entre si, nota-se que conseguiram logo nas primeiras épocas encontrar os padrões necessários para sua correta classificação, logo, em relação a taxa de perdas, vê-se extratos bastante assertivos, conforme gráfico 18 a seguir.

Gráfico 18 - Validação das perdas



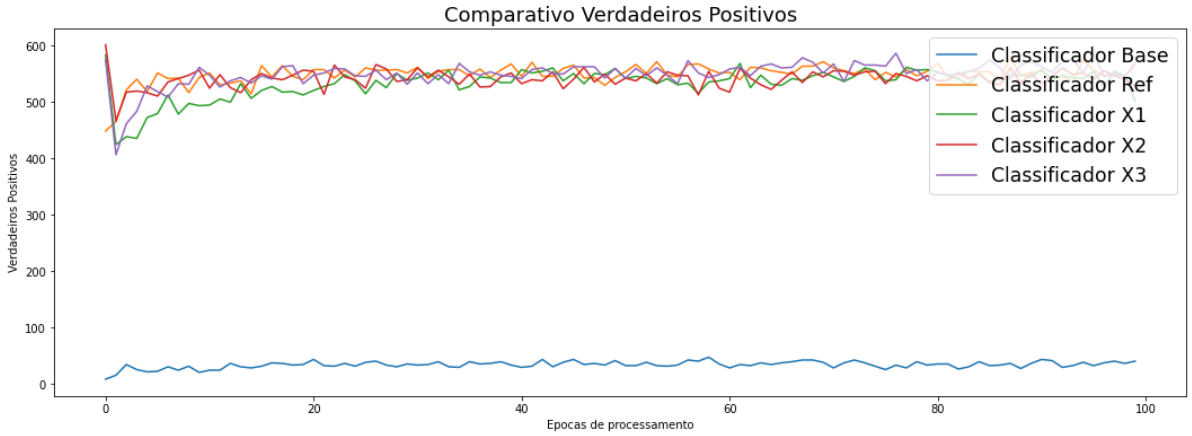
Fonte: Autor, 2022.

Em uma análise complementar, a taxa de perda em fase de validação, nota-se algumas inconsistências encontradas nas primeiras épocas, sendo

resolvidas logo em seguida pela própria CNN.

A seguir serão apresentados os gráficos de variáveis, tais como, verdadeiros positivos (VP) ou (TP), falsos positivos (FP), verdadeiros negativos (VN) ou (TN) e falsos negativos (FN).

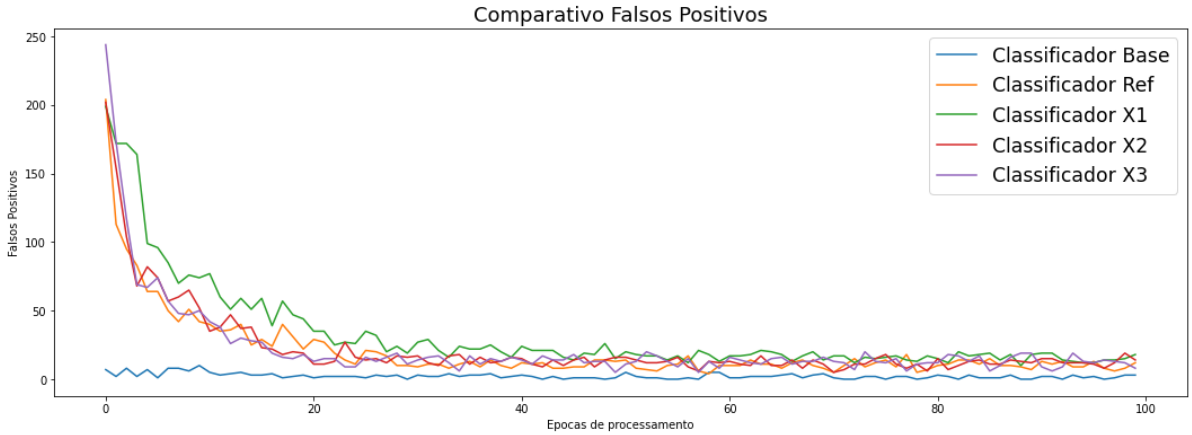
Gráfico 19 - Verdadeiros Positivos



Fonte: Autor, 2022.

Vê-se que os VP, representados no gráfico 19, embora apresentem uma leve queda nas épocas iniciais, os modelos se estabelecem nas épocas finais.

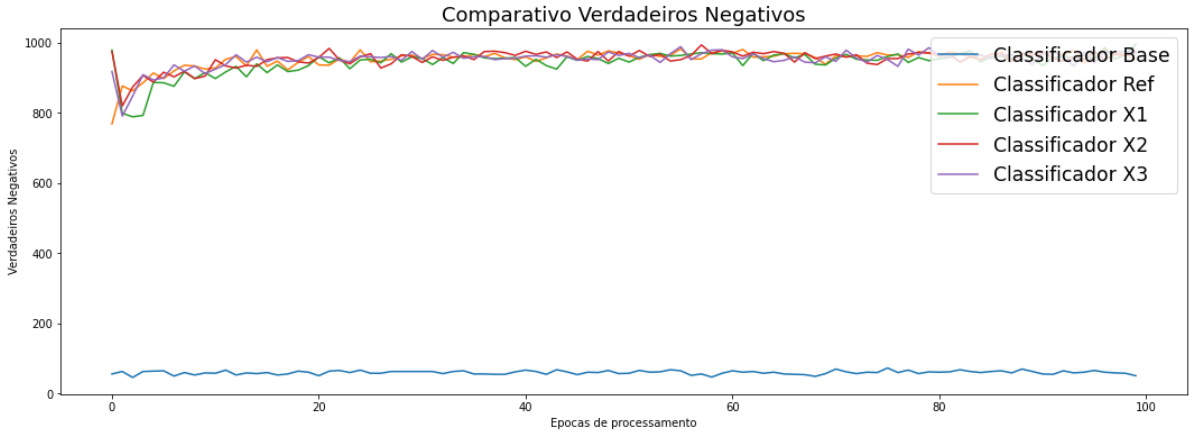
Gráfico 20 - Falsos Positivos



Fonte: Autor, 2022.

Analisando o gráfico 20 que poderia inicialmente dar a ideia de eventuais FP, os mesmos não persistem e logo depois das épocas iniciais eles caem e se estabilizam até o final das 100 épocas.

Gráfico 21 - Verdadeiros Positivos

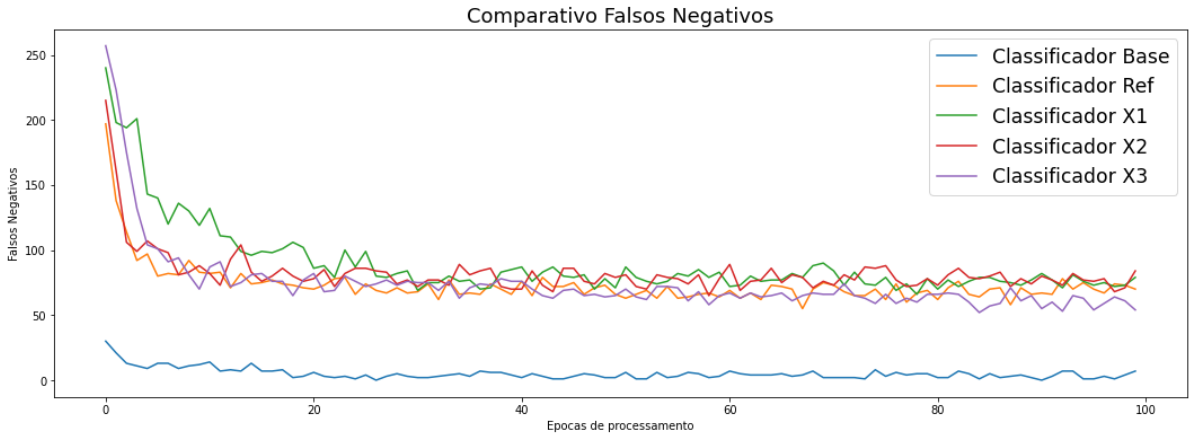


Fonte: Autor, 2022.

Em relação aos VN, no gráfico 21 acima mesmos de mantem estáveis quase todas as épocas em um número considerável.

Já no gráfico 22, a seguir o qual traz os FN a números bem baixos, destacando o modelo X3, conforme os indicadores anteriores.

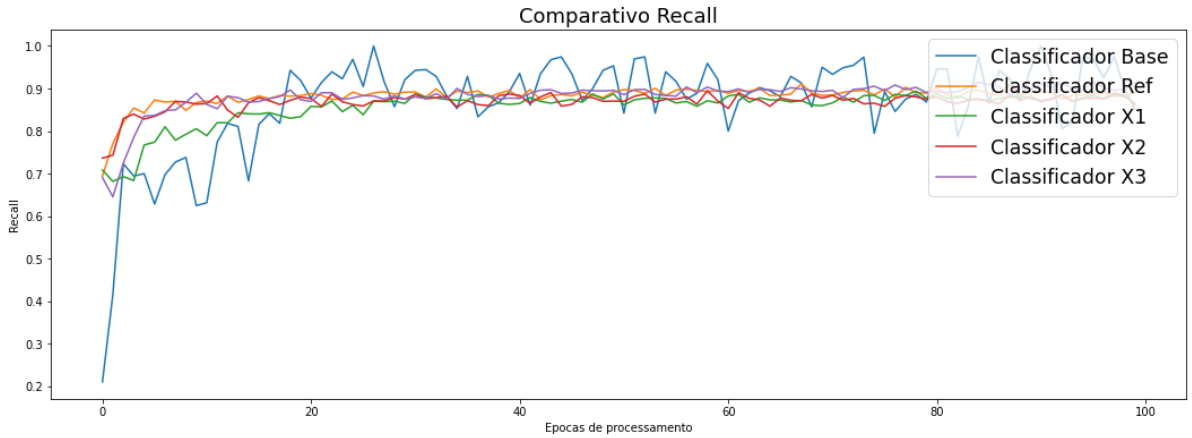
Gráfico 22 - Verdadeiros Positivos



Fonte: Autor, 2022.

Seguindo a análise comparativa entre os modelos o último a ser analisado é o gráfico 23 que apresenta a sensibilidade, sendo a frequência na qual, o classificador detecta exemplos de uma determinada classe.

Gráfico 23 – Sensibilidade

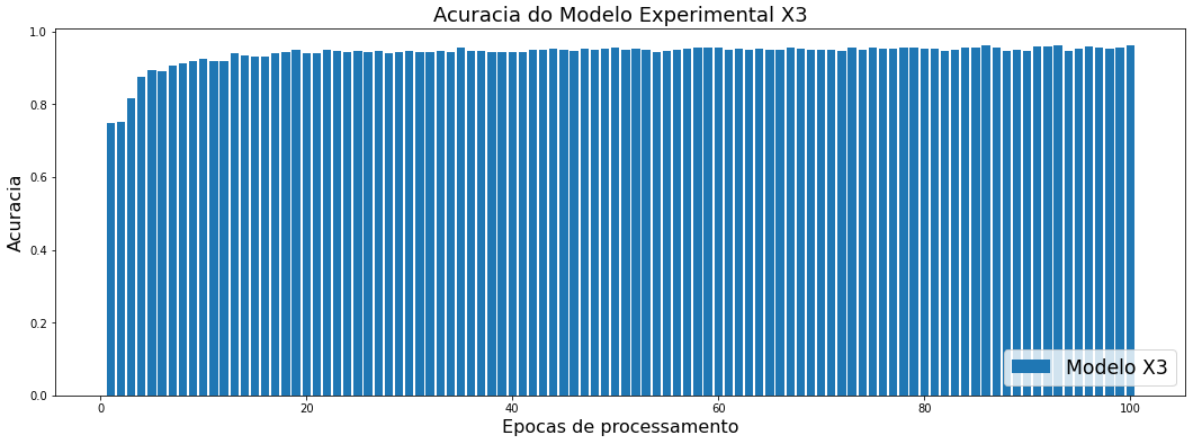


Fonte: Autor, 2022.

Com isso destaca-se o modelo X3, como um modelo mais estável e aplicável, por apresentar melhor resultado que todos os modelos comparados entre si.

Para tanto, a seguir será detalhado as métricas desenvolvidas no modelo para melhor visualização prática no modelo.

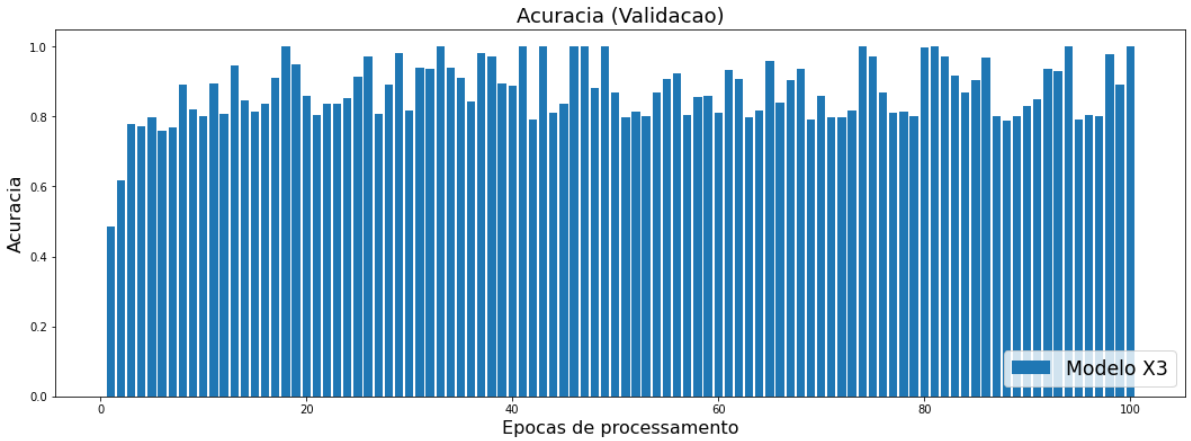
Gráfico 24 - Acurácia do modelo X3



Fonte: Autor, 2022

Ao observar o gráfico 24, a acurácia logo após as épocas iniciais, vê-se uma estabilidade importante em torno de 0,95, atingindo 0,961%, todavia modelagem como esta necessita de validação como será tratado a seguir.

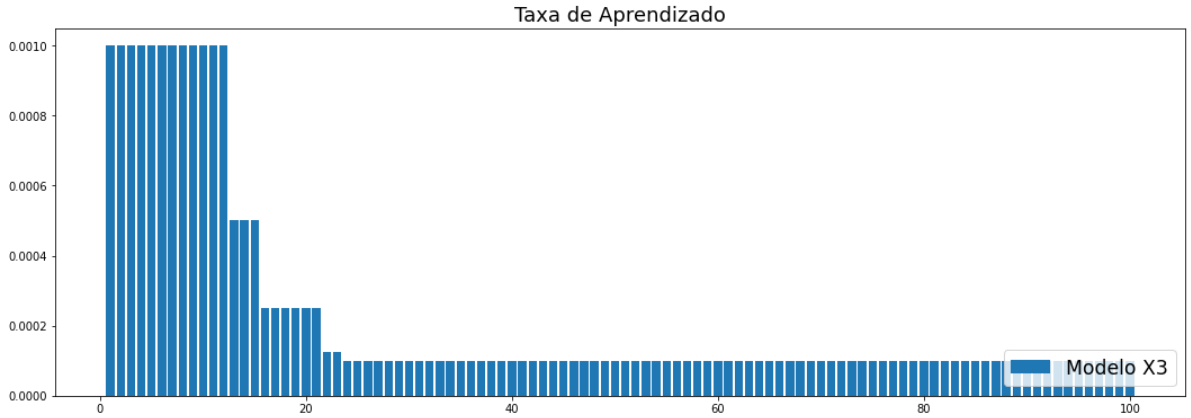
Gráfico 25 - Acurácia Validação modelo X3



Fonte: Autor, 2022.

Já na validação da acurácia, conforme gráfico 25, nota-se que uma completa constatação do próprio modelo trazendo picos de 0,999% e finalizando após 100 épocas os mesmos valores, ou seja, uma performance interessante e relevante da aplicação

Gráfico 26 - Taxa de Aprendizado do modelo X3

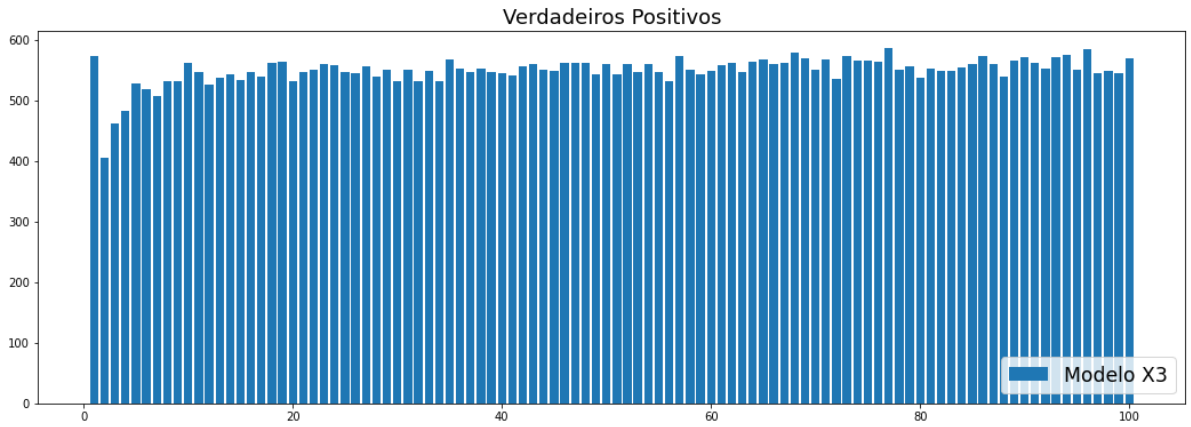


Fonte: Autor, 2022.

Quanto a taxa de aprendizado, visto no gráfico 26, até as 20 épocas iniciais, onde o modelo normaliza e se mantem, encontra-se uma aprendizagem

menor e contínua. Isso demonstra que as convoluções vão e voltam em suas camadas até encontrarem a autoaprendizagem, partindo de conexões e parâmetros da CNN.

Gráfico 27 - Verdadeiros Positivos do modelo X3

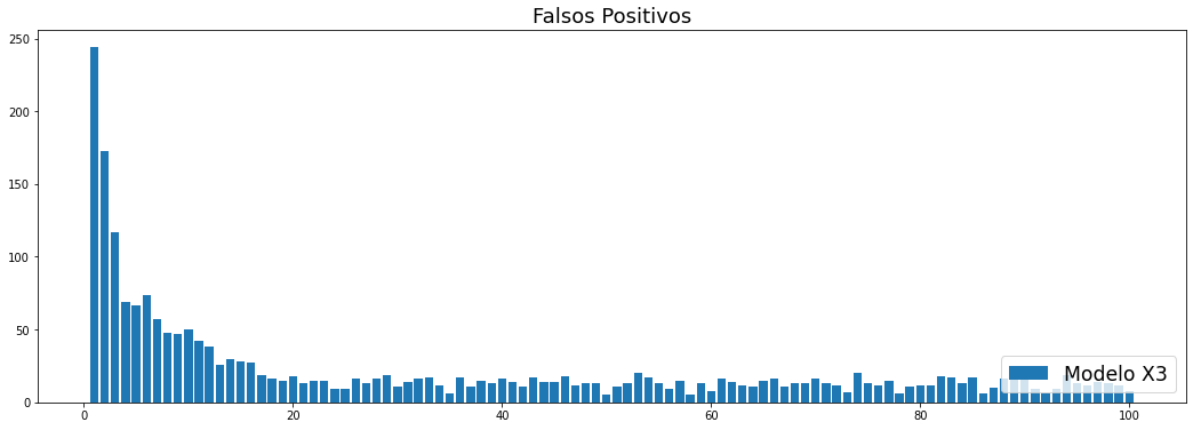


Fonte: Autor, 2022.

O gráfico 27 que trata os VP, pontua uma progressiva e visível estabilidade com números constantes o que retrata um modelo com muitos acertos.

Contrário dos resultados obtidos Noonan (2002), relata em uma pesquisa mais antiga que valores de VP eram destoantes aos acertos em CNN, talvez pela evolução dos modelos atuais

Gráfico 28 - Falsos Positivos do modelo X3

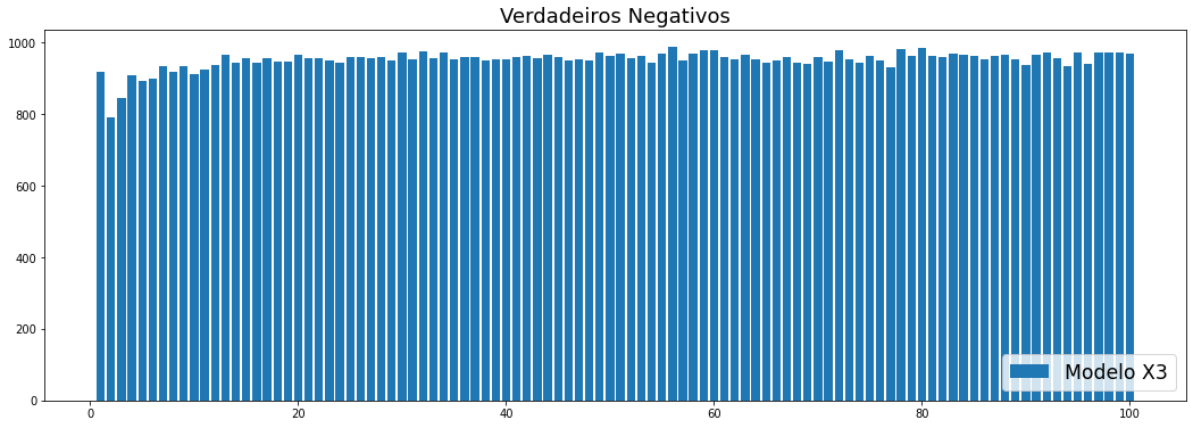


Fonte: Autor, 2022.

Inversamente proporcional, os FP, destacados no gráfico 28, que mapeariam as falhas do modelo, iniciam-se de forma intensa e depois decrescem, estabilizando em valores mínimos diante das épocas finais.

Fato corroborado por Chang *et al.* (2022) que menciona que os FP são capazes de apresentar que a patologia realmente foi identificada.

Gráfico 29 - Verdadeiros Negativos do modelo X3

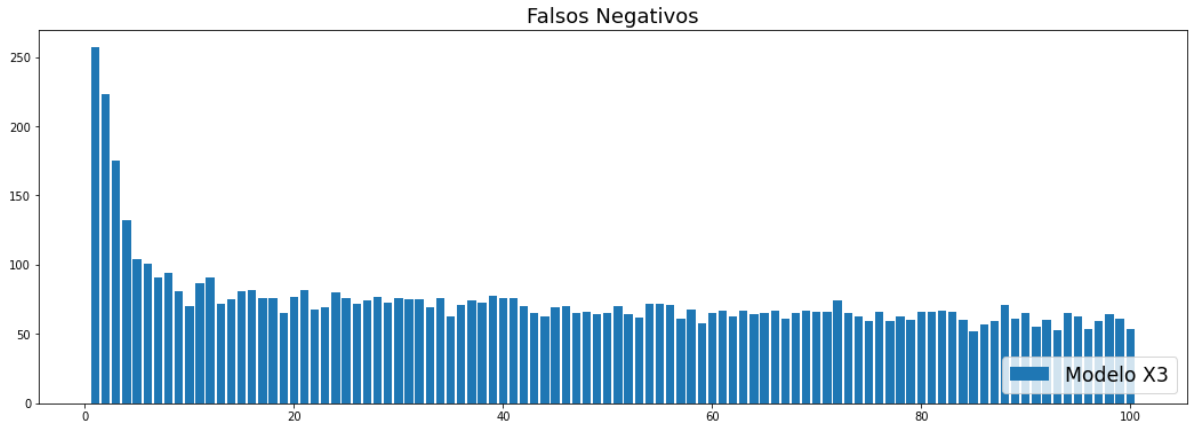


Fonte: Autor, 2022.

Em uma confirmação ao que realmente é negativo, os VN do gráfico 29,

retratam um relativo e constante projeção, mantendo-se estável até as 100 épocas finais.

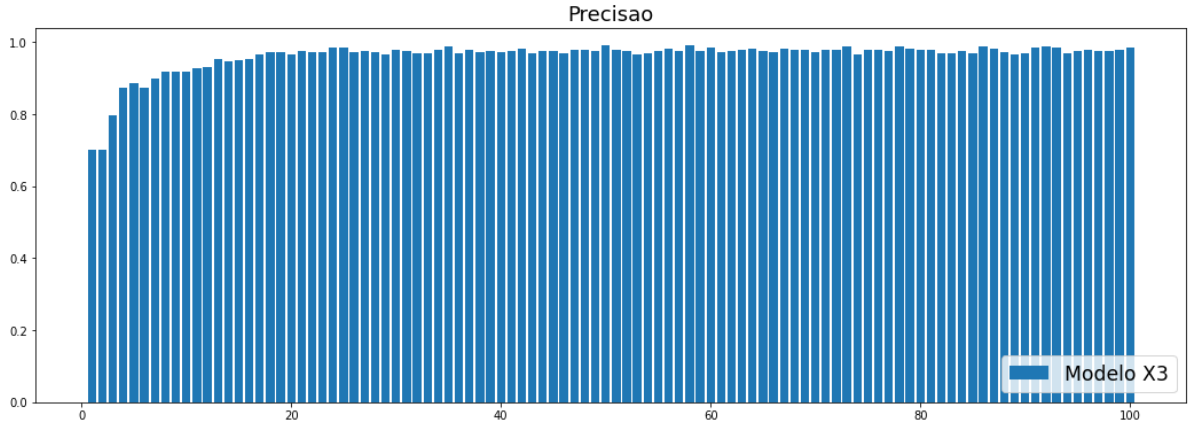
Gráfico 30 - Falsos Negativos do modelo X3



Fonte: Autor, 2022.

Da mesma forma que VN, destacado no gráfico 30, apresenta-se inversamente equilibrado aos VN, retratando os FN em valores contínuos baixos. Fraiwan *et al* (2022), menciona que a taxa de verdadeiro negativo, mede a capacidade de identificar elementos negativos. Sendo assim, estes parâmetros devem ser considerados baixos.

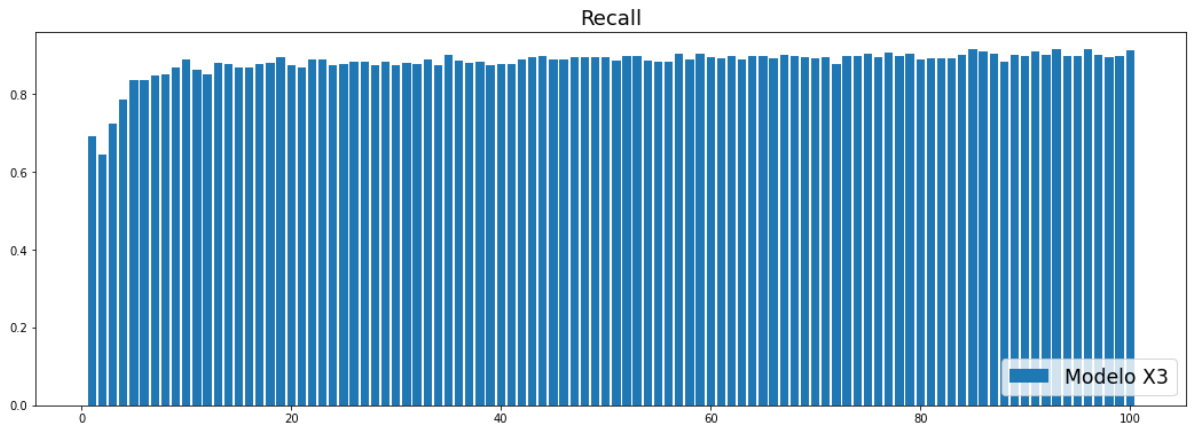
Gráfico 31 - Precisão do modelo X3



Fonte: Autor, 2022

Quanto à precisão do modelo eleito de acordo com o gráfico 31, por seus números tão representativos, a partir das 20 épocas, ou seja, mantém-se estável até ao final com o mínimo de perda, confirmado essa métrica. A precisão informa a proporção de predições positivas que realmente são VP. Quanto maior a quantidade de FP, menor será a precisão do modelo de acordo com Davis e Goadrich (2006)

Gráfico 32 - Sensibilidade do modelo X3



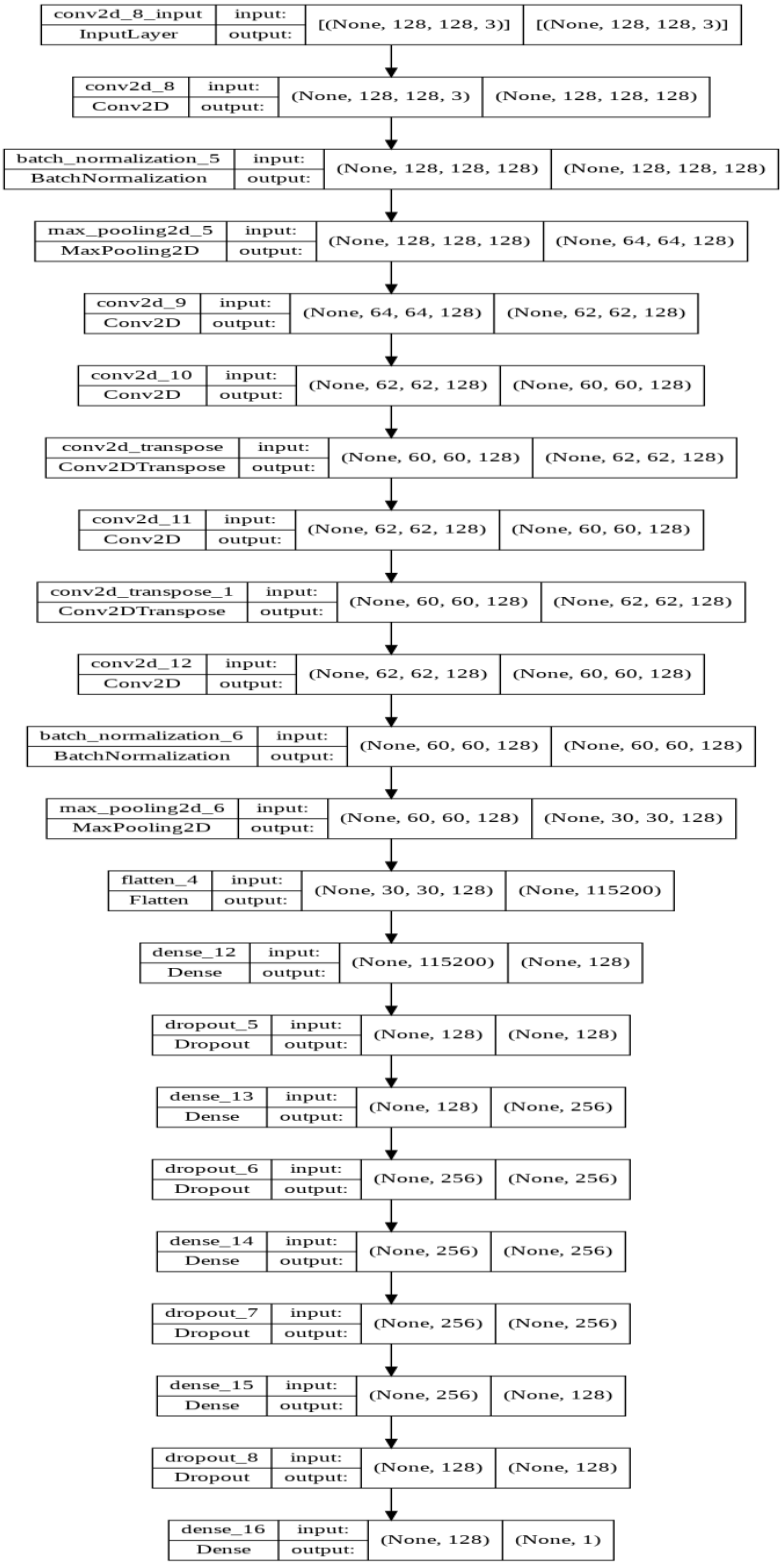
Fonte: Autor, 2022.

Conforme o gráfico 32 a métrica de sensibilidade do modelo se estabelece a partir de 10 épocas, onde ela própria se arranja e na sequência mantem a estabilidade do algoritmo, a qual aponta a proporção dos positivos.

Os estudos para a determinação e identificação de diversas doenças da coluna tem sido aprimorado na atualidade com a utilização da IA, utilizando métricas tais como a precisão, acurácia, sensibilidade e especificidade conforme apontado por Azimi *et al*. (p.558, 2020)

Na sequência destaca-se o sumário de classificação, que transcorre todas as convoluções até a saída final de identificação da imagem de RX, representada na figura 80 a seguir.

Figura 82 - Sumário do Classificador



Fonte: Autor, 2022.

Para tanto, um bloco de códigos referente ao carregamento da imagem a ser testada, recebe um pré-processamento, tendo seu tamanho e formato convertido para uma matriz a ser processada pela CNN, conforme pode ser visualizado na figura 81.

Figura 83 - Entrada de códigos com vistas aos resultados

```
img_teste = load_img('RADIOGRAFIA_0_8131.jpg')

percent = 80
width = int(img_plot1.shape[1] * percent / 100)
height = int(img_plot1.shape[1] * percent / 100)
width = int(img_plot2.shape[1] * percent / 100)
height = int(img_plot2.shape[1] * percent / 100)
dim = (width, height)
img1_pronta = cv2.resize(img_plot1, dim)
img2_pronta = cv2.resize(img_plot2, dim)

imagem = 'RADIOGRAFIA_0_8131.jpg'
imagem2 = 'RADIOGRAFIA_0_8131.jpg2.jpg'

img2 = cv2.imread(imagem2)
thresh2, img_thresh2 = cv2.threshold(img2, 240, 255, cv2.THRESH_BINARY_INV)

img_plot_composta = cv2.subtract(img2_pronta, img1_pronta)
img_plot_composta2 = cv2.addWeighted(img2_pronta, 0.2, img1_pronta, 0.5, 1)
img_plot_composta3 = cv2.addWeighted(img2_pronta, 0.2, img1_pronta, 0.5, 1)

```

Fonte: Autor, 2022.

Em seguida são carregadas as imagens a serem utilizadas como referência, sendo a primeira delas a imagem que será submetida a teste pela RNA, seguido das imagens compatíveis com a mesma que possuem marcações a serem apresentadas ao usuário. Todas as imagens passam por um processo de normalização para que possuam o mesmo tamanho e características na plotagem.

Como pode ser visualizado no código da figura 82.

Figura 84 - Codificação para melhor visualização

```
fig = plt.figure(figsize=(10,20))
ax1 = fig.add_subplot(131)
plt.imshow(img2)
ax2 = fig.add_subplot(132)
plt.imshow(img_plot1)
ax3 = fig.add_subplot(133)
plt.imshow(img_thresh2)
ax4 = fig.add_subplot(231)
plt.imshow(img_plot_composta)
ax5 = fig.add_subplot(232)
plt.imshow(img1_pronta)
ax6 = fig.add_subplot(233)
plt.imshow(img_plot_composta2)
plt.show()

img_teste2 = load_img('RADIOGRAFIA_0_8131.jpg8131.jpg', target_size = (128, 128)
img_teste2 = image.img_to_array(img_teste2)
img_teste2 /= 255
img_teste2 = np.expand_dims(img_teste2, axis = 0)

resultado_teste = model.predict(img_teste2)
resultado_final = resultado_teste

```

Fonte: Autor, 2022.

Também é criada uma pequena codificação na figura 83, referente ao modo como as imagens serão apresentadas, neste caso, exibindo 3 imagens, facilitando a visualização e identificação das características das mesmas por parte do usuário.

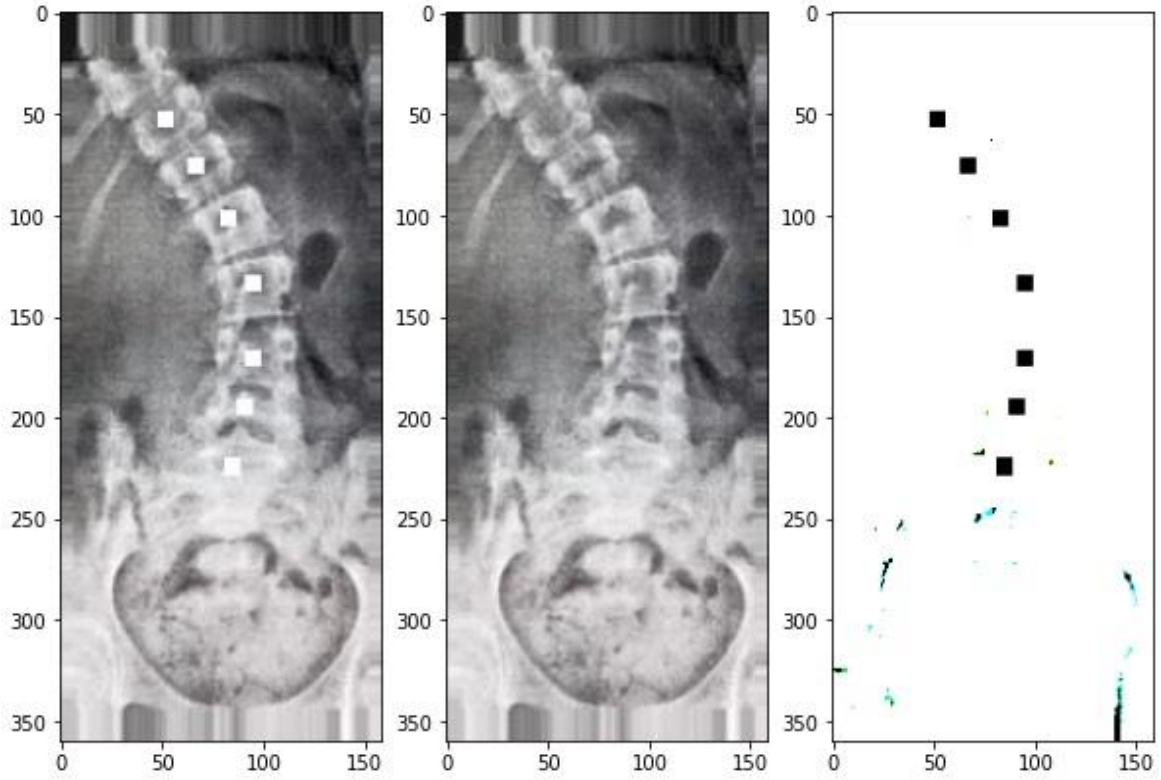
Figura 85 - Carregamento da imagem teste

```
40  img_teste2 = load_img('PATOLOGICA_0_8131.jpeg',
41                        target_size = (128, 128))
42  img_teste2 = image.img_to_array(img_teste2)
43  img_teste2 /= 255
44  img_teste2 = np.expand_dims(img_teste2,
45                              axis = 0)
46
47  resultado_teste = model.predict(img_teste2)
48
49  resultado_final = resultado_teste
50
```

Fonte: Autor, 2022.

Por meio da função *predict( ),* por sua vez foi parametrizada com a variável que carrega a imagem/amostra para testes, onde são aplicados uma série de testes comparando as características de tal imagem com as características de imagens classificadas como normal ou patológica guardadas no aprendizado de máquina, gerando assim um resultado final, classificando a amostra de acordo com suas características, conforme figura 84.

Figura 86 - Identificação do RX de coluna lombar com aspecto de escoliose



Fonte: Autor, 2022.

Executando o código anterior então, é exibido em tela para o usuário, 3 imagens de referência, baseadas na amostra carregada.

Lembrando que das imagens inseridas, as mesmas, passam por uma série de mecanismos internos na CNN de pré e de pós processamento, visando extrair o máximo de características que sejam relevantes para a correta classificação de acordo com a identificação do desvio da coluna.

Figura 87 - Códigos dos resultados de identificação

```
if resultado_final[0] < 0.5:
    print('Radiografia de Coluna Lombar com aspecto NORMAL.')
else:
    print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')

```

```
Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.
```

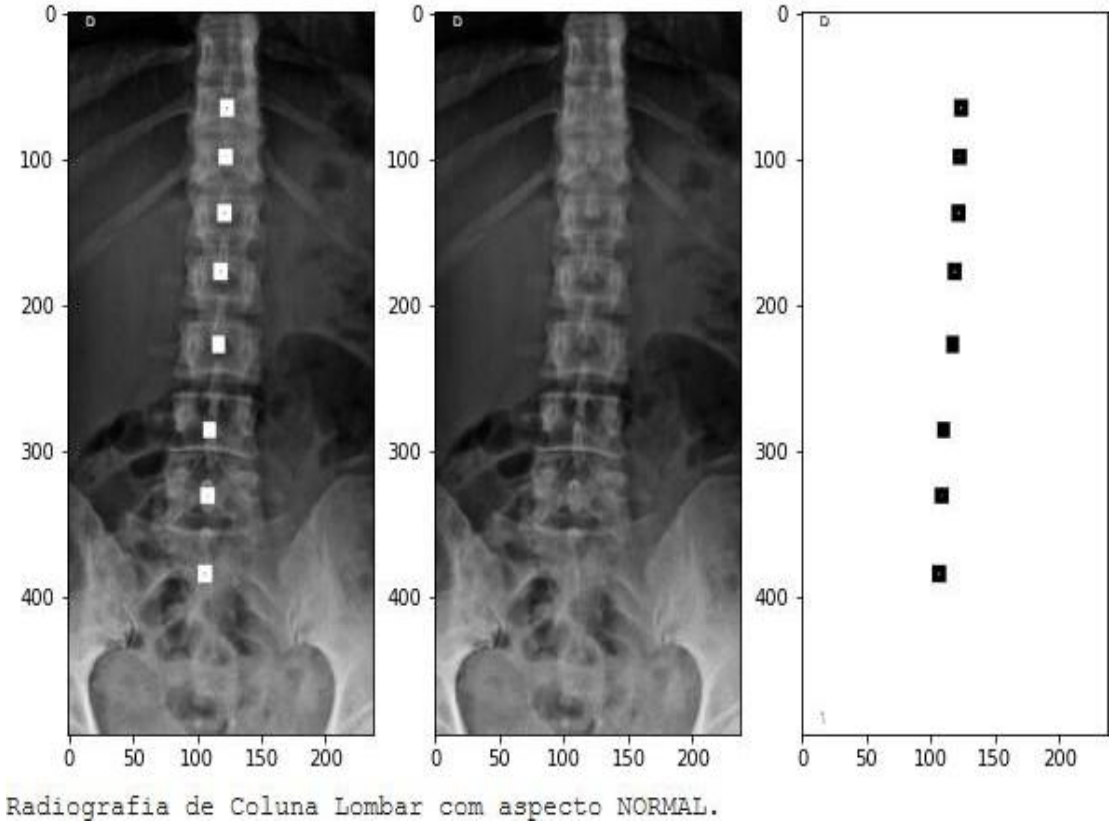
```
if resultado_final[0] < 0.5:
    print('Radiografia de Coluna Lombar com aspecto NORMAL.')
else:
    print('Radiografia de Coluna Lombar com aspecto de ESCOLIOSE.')

```

```
Radiografia de Coluna Lombar com aspecto NORMAL.
```

Fonte: Autor, 2022.

Sendo criado um último bloco conforme figura 85 de código que, dependendo da identificação encontrada para a classificação da amostra, irá exibir em tela um texto explicativo referente ao contexto da classificação.

Figura 88 - Identificação do RX de coluna lombar com aspecto normal

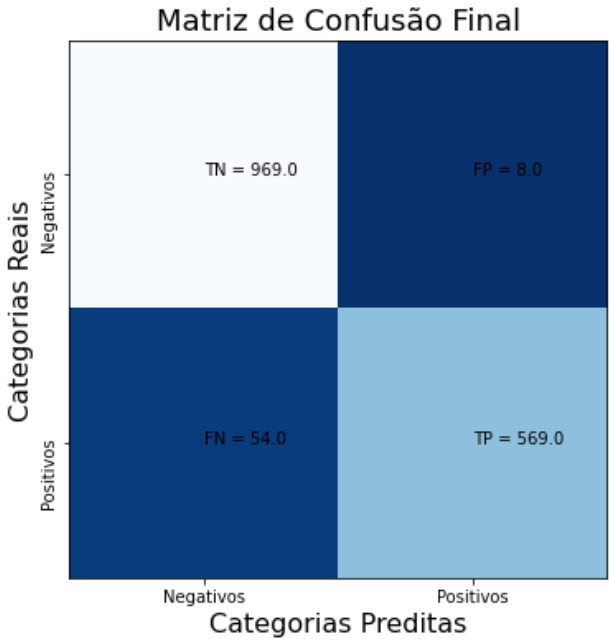


Fonte: Autor, 2022.

A figura 86 reflete um exemplo teste, ou seja, fora aplicado sobre outra amostra, dessa vez de acordo com suas características sendo classificada como normal.

Para tanto, a fim de ratificar os dados colhidos, realizou-se na figura 87 uma Matriz de confusão que representa mais uma forma de confirmação e validação do modelo eleito.

Figura 89 - Matriz de Confusão



Fonte: Autor, 2022.

A partir da Matriz de confusão diversas métricas são obtidas com o propósito de confirmar a validação do modelo, são elas: verdadeiros positivos (VP) ou (TP), falsos positivos (FP), verdadeiros negativos (VN) ou (TN) e falsos negativos (FN), recurso utilizado muito na área da saúde.

Tais resultados são corroborados por Zhang Zhu, Zhao *et al*. (2022), os quais, utilizam dos mesmos métodos como forma de avaliação de performance do modelo, estratificando a confirmação e validação, inclusive avaliando desvios de coluna.

Da mesma forma, Samina *et al*. (2022) também traz o mesmo apontamento de VN, VP, FN e FP na otimização de CNN com imagens de RX.

Todavia, outras métricas se perfazem, cumprindo os dados alcançados dos objetivos propostos elaborados a partir da própria CNN, são elas:

- Acurácia: 0,961
- Precisão: 0,986
- Sensibilidade: 0,913
- Especificidade: 0,991

Entretanto, optou-se por incrementar a análise do algoritmo com outras métricas complementares, tais como, o FMI, o MCC e IY que após seus cálculos, os mesmo representaram respectivamente os seguintes resultados FMI=0,934, MCC=0,894 e IY=0,877, ou seja, a RNA proposta pode refinar-se ainda mais, sob novas perpectivas de estudos e programações futuras. Vilela Júnior *et al*. (2022). Essas métricas são amplamente utilizadas mormente por Wang, Khan e Zhang (2021) onde reforçam os dados da matriz de confusão.

Sendo assim, realizado o aprimoramento da apresentação dos resultados e refinamento das métricas, como observado nos testes realizados, os modelos finais se mostraram bastante eficientes no processo de classificação das imagens, com altos níveis de acurácia, precisão e pouquíssimas perdas obtidas como retorno do processamento dos modelos.

Com a evolução atual dos modelos em *deep learning* e CNN, tem-se resolvido muitos diagnósticos clínicos de forma eficaz, como menciona (FATIMA *et al* 2021).

Desta forma, tais modelos podem ainda mais se superarem em decodificar outras camadas convolucionais em estudos futuros frente às necessidades do cotidiano da área da saúde.

Todos os resultados produzidos são resultados internos de uma RNA,

da criação até a concretização utilizando a IA para identificar desvios escolióticos de coluna lombar, ou seja, o foco do estudo foi concentrado em gerar uma ferramenta a qual o usuário conseguisse inserir no sistema uma imagem de exame, processando a mesma para se obter resultado identificado como “normal” ou “patológico”, e neste processo gerando um retorno didático a partir dos resultados produzidos pelo algoritmo.

6 CONSIDERAÇÕES FINAIS

A hipótese experimental foi corroborada pelos resultados obtidos, destacando a potencialidade de aplicação do presente algoritmo.

Dada a importância e relevância de desenvolvimento tecnológico aplicado na grande área da saúde, especialmente nas CMH, pode-se considerar, depois da criação desse algoritmo, bem como, sua complexa programação de modelos em *machine learning*, frente aos testes, retestes e validações, onde a CNN tem o resultado esperado.

Usualmente a maioria dos estudos referenciados se limitaram a reportar exclusivamente a acurácia da CNN; no presente estudo foram calculadas adicionalmente seis métricas adicionais, que são imprescindíveis para minimização do impacto de resultados VN, FN e FP, sendo um diferencial que contribui com o rigor metodológico na elaboração de Redes Neurais Artificiais na área da saúde. Estas métricas foram: precisão, sensibilidade, especificidade, FMI, MCC e IY, as quais representam uma inovação nas tendências de confirmação e validação de algoritmos, sendo tidos como forma de ratificação dos dados pesquisados e avaliação de performance dos modelos.

Tudo isso, traz um caráter de vanguarda para um estudo oriundo das CMH, pois reúne características multidisciplinares importantes para o conhecimento científico plural e abrangente, possibilitando aos profissionais da área, grande eficiência na interpretação de exames de RX, pois os resultados são considerados considerado satisfatórios e viáveis na aplicação pré diagnóstica.

Portanto, cabe considerar mormente que em virtude da importante eficiência do algoritmo e possibilidades diversas de sua aplicação profissional da

área da saúde, reabilitação e promoção da saúde, ortopedia em geral, clínicas de diagnósticos de imagem, consultórios fisioterápicos, estúdios de *personal trainner*, hospitais e por fim empresas e/ou fabricantes de aparelhos de Raio X.

Diante do exposto, o conhecimento técnico científico, caminha a passos largos no sentido de um futuro com dispositivos e interfaces mais intuitivas para a humanidade, onde outros algoritmos inteligentes complementarão a vida das pessoas e organizações, seguindo em evolução, destacando-se diante de um cenário eminente de transformações e complexidades oriundas do movimento humano e de sua grandeza.

REFERÊNCIAS

ABDOU, M. A. Literature review: Efficient deep neural networks techniques for medical image analysis. **Neural Computing and Applications**, p. 1-22, 2022.

ACM. **Proceedings of the 23rd international conference on Machine learning**.

ALAUDDIN, M. S.; BAHARUDDIN, A. S.; MOHD GHAZALI, M. I. The modern and digital transformation of oral health care: A mini review. In: Healthcare. **Multidisciplinary Digital Publishing Institute**, 2021. p. 118.

ASRI, E. *et al.* Detection of Mask Usage Using Image Processing and Convolutional Neural Network (CNN) Methods. **Jurnal Teknologi Informasi dan Pendidikan**, 15(1), 1-10, 2022.

AZIMI, P. *et al*. A Review on the Use of Artificial Intelligence in Spinal Diseases. **Asian Spine Journal** 14, 543–571, 2020. doi:10.31616/asj.2020.0147.

BAKA, N. *et al*. "Ultrasound Aided Vertebral Level Localization for Lumbar Surgery." **IEEE Trans Med Imaging** 36(10): 2138-2147, 2017

BELHARBI, S. *et al*. "Spotting L3 slice in CT scans using deep convolutional network and transfer learning." **Comput Biol Med** 87: 95-103, 2017

BIZZOCA, D. *et al*. Anterior vertebral body tethering for idiopathic scoliosis in

growing children: A systematic review. **World Journal of Orthopedics**, v. 13, n. 5, p. 481-493, 2022.

CASAGRANDE, R. M. *et al*. Determining the three-dimensional position of the acetabular cup in total hip arthroplasty patients. **Revista CPAQV**, v. 7, n. 3, 2015.

CHANG *et. al*. Automated detection and segmentation of sclerotic spinal lesions on body CTs using a deep convolutional neural network. **Skeletal Radiol**. 2022 Feb;51(2):391-399. doi: 10.1007/s00256-021-03873-x. Epub 2021 Jul 22. PMID: 34291325.

CHOU R, *et al*. Noninvasive Treatments for Low Back Pain [Internet]. Rockville (MD): **Agency for Healthcare Research and Quality** (US); 2016 Feb. Report No.: 16-EHC004-EF. PMID: 26985522.

DAVIS, J.; GOADRICH, M. **The relationship between precision-recall and roc curves**. In: [S.l.], 2006. p. 233–240.

DEUTSCH, J. E.; GILL-BODY, K. M.; SCHENKMAN, M. Updated Integrated Framework for Making Clinical Decisions across the Lifespan and Health Conditions. **Physical Therapy**. 2022.

FAN, G. *et al*. "**Deep learning-based lumbosacral reconstruction for difficulty prediction of percutaneous endoscopic transforaminal discectomy at L5/S1 level**: A retrospective cohort study." Int J Surg 82: 162-169, 2020.

FATIMA, J. *et al.* Spinal vertebrae localization and analysis on disproportionality in curvature using radiography—a comprehensive review. **EURASIP Journal on Image and Video Processing,** 2021. doi:10.1186/s13640-021-00563-5

FRAIWAN, *et. al*. Using deep transfer learning to detect scoliosis and spondylolisthesis from x-ray images. **PLOS ONE** 17, e0267851, 2022. doi: 10.1371/journal.pone.0267851

FRANÇA, F. R. *et al.* Segmental stabilization and muscular strengthening in chronic low back pain: a comparative study. **Clinics**, 65(10), 1013-1017, 2010.

FRIEDMAN, M., FRIEDLAND, G. W. **Medicine's 10 Greatest Discoveries**. Yale University Press, 2000

HOSSAIN, M. B. *et al.* Transfer learning with fine-tuned deep CNN ResNet50 model for classifying COVID-19 from chest X-ray images. **Informatics in Medicine Unlocked**, 30, 100916, 2022.

JACHSTET, *et al.* Prevalência de dores nas costas e fatores de risco relacionados ao estilo de vida de escolares do Rio Grande do Sul. **Brazilian Journal of Development**. Curitiba. Vol. 8, n. 2, Feb. 2022, p. 9876-9888.

KAUR, R.; BHATTACHARYA, J.; CHANA, I. Deep CNN based online image deduplication technique for cloud storage system. **Multimedia Tools and**

Applications, 2022, p. 1-34.

KAUSCH, L. *et al*. "Toward automatic C-arm positioning for standard projections in orthopedic surgery." **Int J Comput Assist Radiol Surg** 15(7): 1095-1105, 2020

KERAS. Disponível em: https://keras.io/api/ Acesso em: 25 abril 2022.

LADDHA, S.; KUMAR, V. DGCNN: deep convolutional generative adversarial network based convolutional neural network for diagnosis of COVID-19. **Multimedia Tools and Applications**, 2022, p. 1-18.

LAKHANI, P.; SUNDARAM, B. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. **Radiology**, v. 284, n. 2, p. 574-582, 2017.

LENZ, M., *et. al*. Scoliosis and Prognosis—a systematic review regarding patient-specific and radiological predictive factors for curve progression. **European Spine Journal,** 30(7), 1813-1822, 2021.

LEONARD, H. *et al*. **CDKL5 deficiency disorder**: clinical features, diagnosis, and management. The Lancet Neurology, 2022.

LI, X.; ZHU, D.; DONG, M. Multinomial classification with class-conditional overlapping sparse feature groups. **Pattern Recognition Letters**, 101, 37-43, 2018.

LÖFFLER, M. T. *et al*. "Opportunistic Osteoporosis Screening Reveals Low Bone Density in Patients With Screw Loosening After Lumbar Semi-Rigid Instrumentation: A Case-Control Study." **Front Endocrinol** (Lausanne) 11: 552719, 2020.

MATPLOTLIB. Disponível em: https://matplotlib.org/2.0.2/api/pyplot_api.html Acesso em: 25 abril 2022.

MCAVINEY, J. *et al*. The prevalence of adult de novo scoliosis: A Lsystematic review and meta-analysis. **Eur Spine J 29**, 2960–2969, 2020. https://doi.org/10.1007/s00586-020-06453-0

MORLEY, J. *et al.* The ethics of AI in health care: a mapping review. **Social Science & Medicine**, 260, 113172, 2020.

NEGRINI, S. *et al*. The classification of scoliosis braces developed by SOSORT with SRS, ISPO, and POSNA and approved by ESPRM. **European Spine Journal**, p. 1-10, 2022.

NOH, S. H. Epidemiology of senile spinal diseases: a study based on the Health Insurance Review & Assessment Service database. **Journal of the Korean Medical Association/Taehan Uisa Hyophoe Chi**, 64(3), 2021.

NOONAN K. *et al*. Factors related to false- versus true-positive neuromonitoring

changes in adolescent idiopathic scoliosis surgery. **Spine (Phila Pa 1976)**. 2002. Apr 15;27(8):825-30. doi: 10.1097/00007632-200204150-00009. PMID: 11935104.

NUMPY. Disponível em: https://numpy.org/doc/ Acesso em: 25 abril 2022.

OPENCV. Módulos OpenCV. Disponível em: https://docs.opencv.org/4.5.2/ Acesso em: 25 abril 2022.

PASSOS, C. N. Transformação Digital na Saúde: Desafios e Perspectivas. **Revista Científica Hospital Santa Izabel**, v. 3, n. 3, p. 178-184, 2019.

PYTORCH. Disponível em: https://pytorch.org/docs/stable/index.html Acesso em: 25 abril 2022.

REDDY, S. *et al.* A governance model for the application of AI in healthcare. **Journal of the American Medical Informatics Association**, 27(3), 491-497, 2020.

SALLES, M. C. M. S. Transformação digital em tempos de pandemia. **Revista Estudos e Negócios Academicos**, v. 1, n. 1, p. 91-100, 2021.

SAMINA A., *et. al*. "Optimizing Convolutional Neural Networks with Transfer Learning for Making Classification Report in COVID-19 Chest X-Rays Scans", **Scientific Programming**, Article ID 5145614, 13 p., 2022. https://doi.org/10.1155/2022/5145614

SCHWING, A.G.; URTASUN, R. **Fully connected deep structured networks**. arXiv preprint arXiv:1503.02351, 2015.

SILVA, V. K. S. **Análise de acurácia do diagnóstico assistido por computador no diagnóstico de lesões radiolucentes maxilofaciais-uma revisão sistemática e meta-análise.** 2020.

SOUSA, P. M. *et al*. COVID-19 classification in X-ray chest images using a new convolutional neural network: CNN-COVID. **Research on Biomedical Engineering**, 38(1), 87-97, 2022.

TAGUCHI, T.; NAKANO, S.; NOZAWA, K. Effectiveness of pregabalin treatment for neuropathic pain in patients with spine diseases: a pooled analysis of two multicenter observational studies in Japan. **Journal of Pain Research**, 14, 757, 2021.

TIWARI, V.; SINGHAL, A.; DHANKHAR, N. Detecting COVID-19 Opacity in X-ray Images Using YOLO and RetinaNet Ensemble. In **2022 IEEE Delhi Section Conference (DELCON)** (pp. 1-5). IEEE, 2022

UEDA, A. **O futuro da inovação digital**. MIT Technology Review. 2022. Disponível em: https://mittechreview.com.br/o-futuro-da-inovacao-digital/ Acesso em: 25 abril 2022.

VILELA JUNIOR, G. B. Reflexões e refrações epistemológicas nas ciências do

movimento humano. **Revista CPAQV–Centro de Pesquisas Avançadas em Qualidade de Vida**, Vol 7(2), 2, 2015.

VILELA JUNIOR, G. B. *et al*. Importância do índice fowlkes-mallows (fmi), do coeficientede correlação de matthews (mcc) e do índice youden (iy) nos classificadores de inteligência artificial na área da saúde. **Revista CPAQV** v. 14, n. 3, 2022 > Disponível em: http://www.cpaqv.org/revista/CPAQV/ojs-2.3.7/index.php?journal=CPAQV&page=article&op=view&path%5B%5D=371&path%5B%5D=260 Acesso em: 02 agosto 2022

VILELA JUNIOR, G. B. Você está preparado para a educação 5.0? **Revista CPAQV** v. 12, n. 1 (2020) > Disponível em: <http://www.cpaqv.org/revista/CPAQV/ojs-2.3.7/index.php?journal=CPAQV&page=article&op=view&path%5B%5D=371&path%5B%5D=260> Acesso em: 25 abril 2022.

WANG SH; KHAN MA; ZHANG YD. VISPNN: VGG-inspired Stochastic Pooling Neural Network. **Comput Mater Contin**. 2022;70(2):3081-3097. doi: 10.32604/cmc.2022.019447. Epub 2021 Sep 27. PMID: 35615529; PMCID: PMC7612766.

WANG, F.; PREININGER, A. AI in health: state of the art, challenges, and future directions. **Yearbook of medical informatics**, 28(01), 016-026, 2019.

WASINPONGWANICH, K.; NOPSOPON, T.; PONGPIRUL, K. **Surgical**

Treatments for Lumbar Spine Diseases: A Systematic Review and Meta-Analysis. medRxiv, 2021.

WONG, L. PK; CHEUNG, P. WH; CHEUNG, J. PY. Curve type, flexibility, correction, and rotation are predictors of curve progression in patients with adolescent idiopathic scoliosis undergoing conservative treatment: a systematic review. **The Bone & Joint Journal**, v. 104, n. 4, p. 424-432, 2022.

WORLD HEALTH ORGANIZATION. **ICD-11 for mortality and morbidity statistics**. Version: 2022 April. Geneva: WHO; 2022. Disponível em: https://icd.who.int/en Acesso em: 25 abril 2022

WU, A. *et al.* Global low back pain prevalence and years lived with disability from 1990 to 2017: estimates from the Global Burden of Disease Study 2017. **Annals of translational medicine**, 8(6), 299, 2020. Disponível em: https://doi.org/10.21037/atm.2020.02.175 Acesso em: 25 abril 2022

YASAKA, K., *et al*. "Prediction of bone mineral density from computed tomography: application of deep learning with a convolutional neural network." **Eur Radiol** 30(6): 3549-3557, 2020.

YAŞAR, H.; CEYLAN, M. A novel study for automatic two-class COVID-19 diagnosis (between COVID-19 and Healthy, Pneumonia) on X-ray images using texture analysis and 2-D/3-D convolutional neural networks. **Multimedia Systems**, 1-19, 2022.

ZHANG T. *et al*. A clinical classification for radiation-less monitoring of scoliosis based on deep learning of back photographs. **Research Square**; (2022). DOI: 10.21203/rs.3.rs-1655808/v1.

ZHANG, T. *et al*. "A novel tool to provide predictable alignment data irrespective of source and image quality acquired on mobile phones: what engineers can offer clinicians." **Eur Spine J** 29(3): 387-395, 2020.

ZHOU, Y. *et al*. "Automatic Lumbar MRI Detection and Identification Based on Deep Learning." **J Digit Imaging** 32(3): 513-520, 2019

APÊNDICES

Termo de confidencialidade e cessão de dados para pesquisa

Centro de Pesquisas Avançadas em Qualidade de Vida

Núcleo de Pesquisas em Biomecânica Ocupacional e Qualidade de Vida

UNIMEP - Universidade Metodista de Piracicaba

TERMO DE CONFIDENCIALIDADE E CESSSÃO DADOS PARA PESQUISA

Declaramos manter a confidencialidade, como também, não haver conflito de interesses de qualquer natureza entre a **Dra. Eloa Regina Gusso**, portadora do CRM 20614 PR - RQE 1443, brasileira, **Guanis de Barros Vilela Junior**, portador do RG: 9597561 brasileiro; **José Ricardo Lourenço de Oliveira**, portador do RG: 18 900255-4, brasileiro; **Ricardo Pablo Passos**, portador do RG: 52264558-6 brasileiro, no que se refere à cessão de dados médicos (radiografias e laudos) para pesquisa em nível de doutorado, junto a Universidade Metodista de Piracicaba, que serão fornecidos pelo Instituto Forlanini, por meio de sua responsável técnica, respaldados pela LGPD - Lei Geral de Proteção de Dados.

Curitiba, PR, 05 de março de 2021.

Dra. Eloa Regina Gusso

Eloa Regina Gusso
ASSINADO
eloagusso@hotmail.com
IP: 179.186.186.70
Assinado em: 06/03/2021, 5:31:07

Me. José Ricardo Lourenço de Oliveira

JOSE RICARDO LOURENCO OLIVEIRA
ASSINADO
contudo@gmail.com
IP: 189.123.244.189
Assinado em: 06/03/2021, 7:41:24

Me.Ricardo Pablo Passos

Ricardo Pablo Passos
ASSINADO
rppasso@gmail.com
IP: 187.106.32.240
Assinado em: 24/05/2022, 2:21:27

Prof. Dr. Guanis de Barros Vilela junior

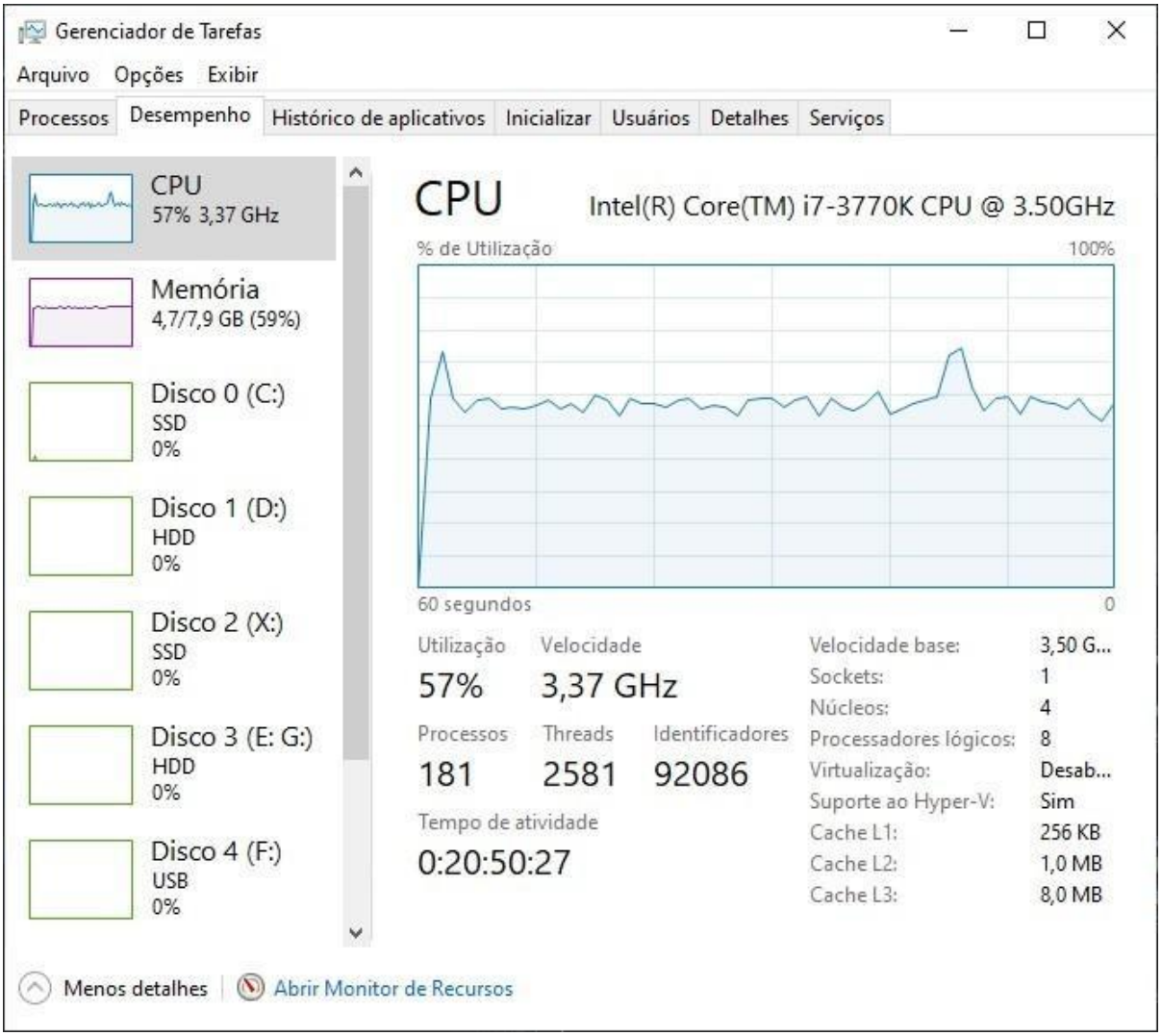
Guanis de Barros Vilela Junior
ASSINADO
guanis@gmail.com
IP: 187.106.32.240
Assinado em: 24/05/2022, 2:23:27

1. Hardware utilizado - computador local

Com a finalidade e evidenciar os dispositivos utilizados para o desenvolvimento da CNN, retrata-se abaixo toda estrutura empregada na mesma, como segue na figura 1 e 2:

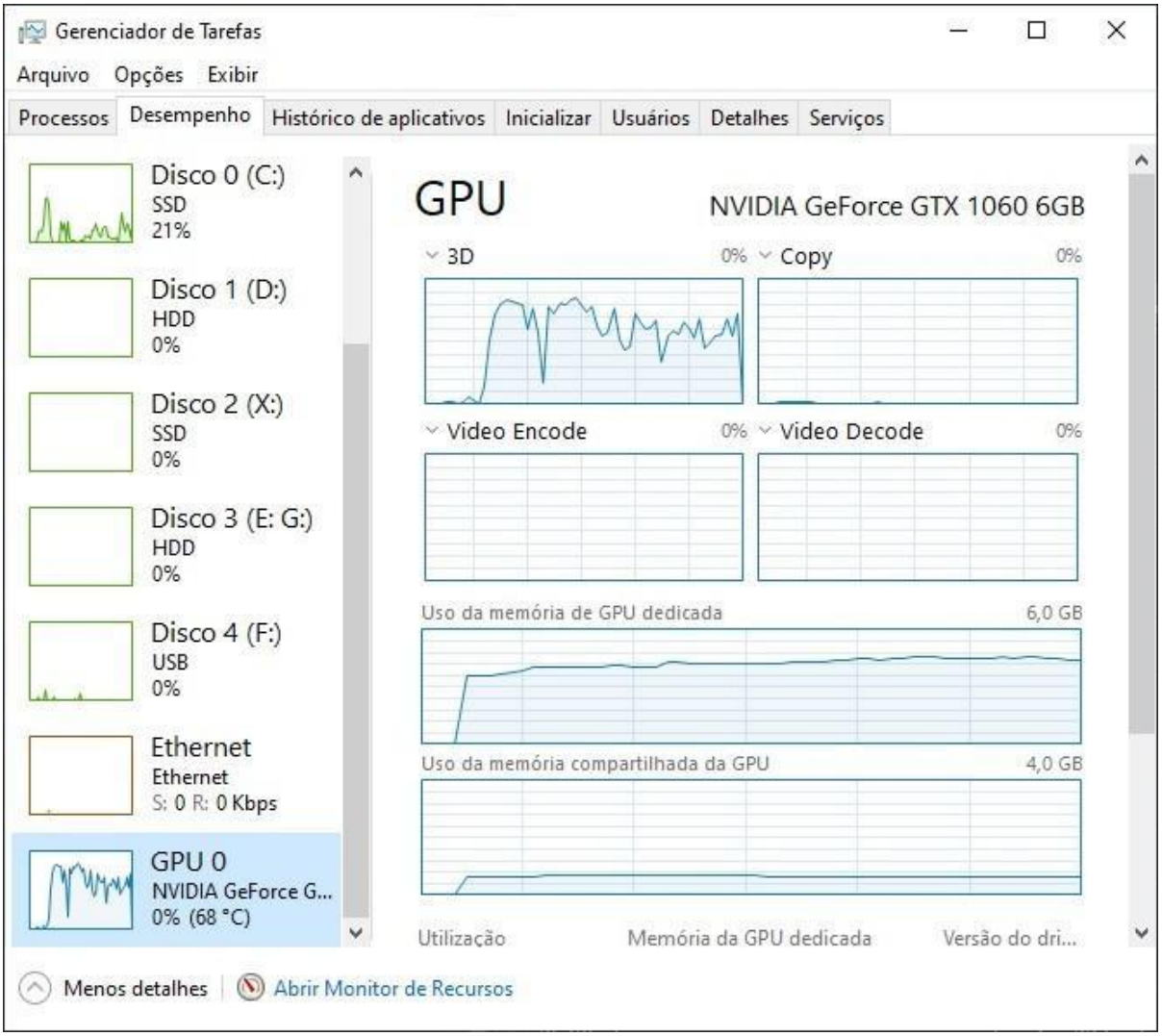
CPU: Intel Core i7 3770k 3.50Ghz 4 núcleos GPU:
nVidia GeForce GTX 1060 6Gb
RAM: 8Gb

Figura 01 - Tela gerenciador de tarefas do computador utilizado – CPU



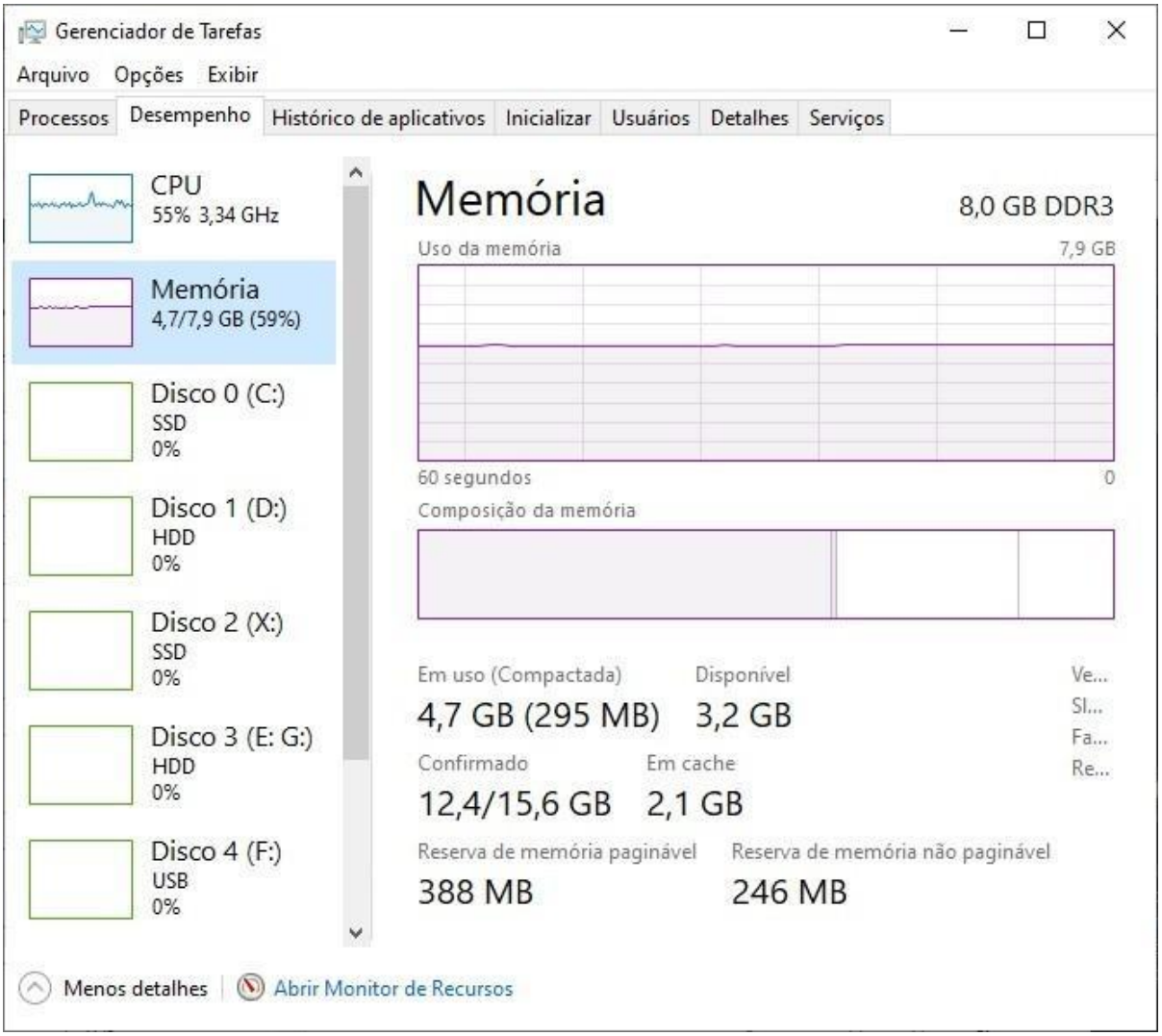
Fonte: Autor, 2022.

Figura 02 - Tela gerenciador de tarefas do computador utilizado - GPU



Fonte: Autor, 2022.

Figura 03 - Tela gerenciador de tarefas do computador utilizado – Memória



Fonte: Autor, 2022.

2.1 Hardware utilizado em nuvem - virtualizado

Para processar os modelos baseados em arquitetura Deep Learning, foi utilizado o processamento em nuvem fornecido pelo *Google Cloud* e para esta aplicação em nuvem foi utilizado um dispositivo com a seguinte configuração:

GPU: 1xTesla K80, compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM

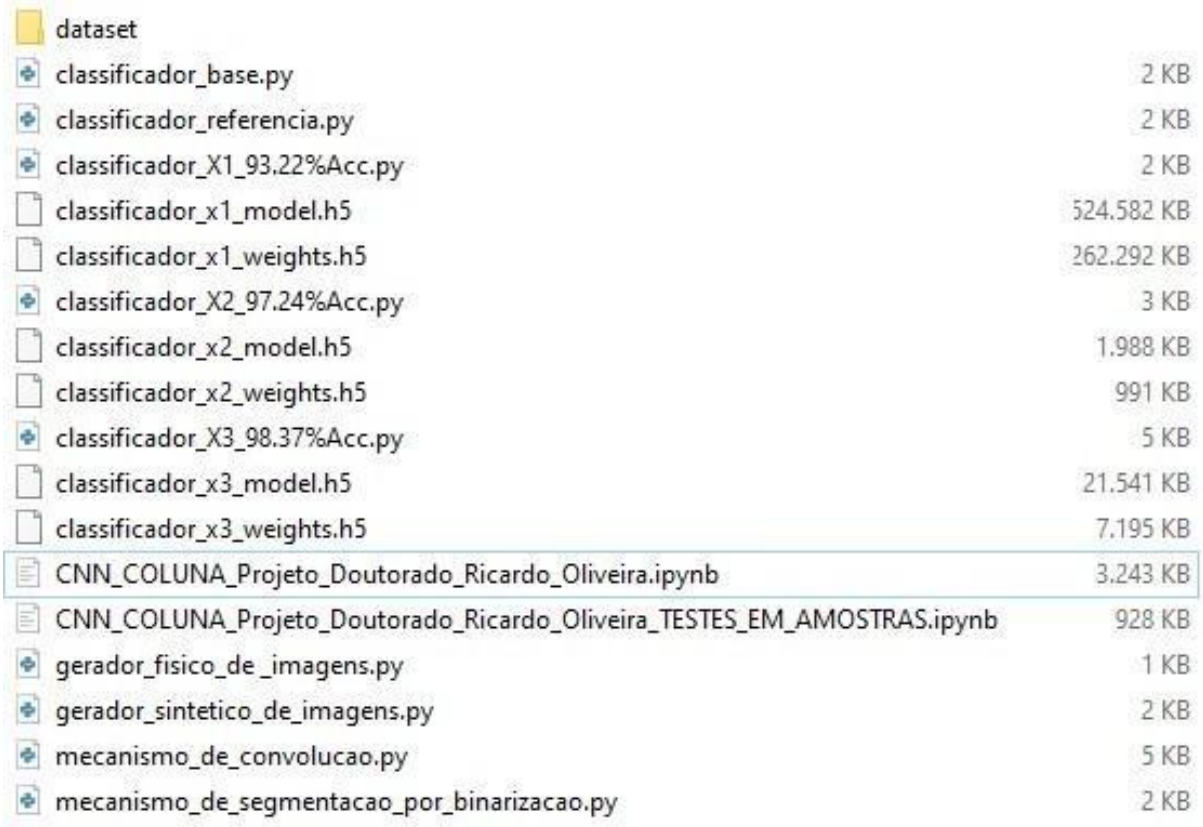
CPU: 1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)

RAM: ~12.6 GB

Mais detalhes sobre a máquina podem ser encontrados em: https://colab.research.google.com/drive/151805XTDg--dgHb3-AXJCpnWaqRhop_2.

Ademais, todo o pacote final de arquivos recebeu nominação e identificação de pastas a seguir demonstradas na figura 4

Figura 4 - Pacote final de arquivos



Fonte: Autor, 2022.

Todos os arquivos gerados para o desenvolvimento da rede neural estão disponíveis na seguinte correspondência eletrônica disponível publicamente na internet

https://drive.google.com/file/d/1OU-bQx5xnsHWi4igdjvIxDOQ8cZy52w-/view?usp=sharing

ANEXOS

1. Parecer consubstanciado do Comitê de Ética e Pesquisa - CEP

UNIMEP
COMITÊ DE ÉTICA EM PESQUISA COM HUMANOS

UNIVERSIDADE METODISTA DE PIRACICABA - UNIMEP

Plataforma Brasil

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: MÉTODOS DA INTELIGÊNCIA ARTIFICIAL APLICADOS NA ANÁLISE DO MOVIMENTO HUMANO

Pesquisador: GUANIS DE BARROS VILELA JUNIOR

Área Temática:

Versão: 1

CAAE: 33912120.9.0000.5507

Instituição Proponente: INSTITUTO EDUCACIONAL PIRACICABANO DA IGREJA METODISTA

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 4.126.546

Apresentação do Projeto:

Projeto adequadamente apresentado, contendo todos os dados necessários para sua análise.

Objetivo da Pesquisa:

Objetivos claros, coerentes com o desenho do projeto e exequíveis dentro do cronograma exposto.

Avaliação dos Riscos e Benefícios:

Os riscos aos sujeitos são mínimos e o projeto assegura o cuidado para reduzi-los. Os benefícios (diretos e indiretos) aos sujeitos estão presentes e superam os riscos.

Comentários e Considerações sobre a Pesquisa:

Destacam-se a relevância e as contribuições da pesquisa apresentada. As bases teóricas estão adequadas, a metodologia é coerente e a coleta de dados é adequada à proposta.

Considerações sobre os Termos de apresentação obrigatória:

O TCLE apresenta as informações necessárias. Acrescentar no início do texto a frase: Você está sendo convidado(a) a participar de um estudo na área da Ciências do Movimento Humano.

Conclusões ou Pendências e Lista de Inadequações:

O projeto está aprovado.

Considerações Finais a critério do CEP:

Este colegiado acolhe o parecer acima descrito e aprova o projeto.

Endereço: Rodovia do Açúcar, Km 156
Bairro: Taquaral **CEP:** 13.400-911
UF: SP **Município:** PIRACICABA
Telefone: (19)3124-1513 **Fax:** (19)3124-1515 **E-mail:** comitedeetica@unimep.br