UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MÁRIO VICTOR GONZAGA MONTEIRO

REDES NEURAIS CONVOLUCIONAIS EXPLICÁVEIS PARA DETECÇÃO DE DOENÇAS PULMONARES UTILIZANDO IMAGENS DE RAIOS-X DO TÓRAX

FORTALEZA

2024

MÁRIO VICTOR GONZAGA MONTEIRO

REDES NEURAIS CONVOLUCIONAIS EXPLICÁVEIS PARA DETECÇÃO DE DOENÇAS PULMONARES UTILIZANDO IMAGENS DE RAIOS-X DO TÓRAX

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Ceará como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Fabrício Gonzalez Nogueira

Coorientador: Prof. Dr. Victor Hugo Costa de Albuquerque

FORTALEZA

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M778r Monteiro, Mário Victor Gonzaga.
Redes Neurais Convolucionais Explicáveis para detecção de doenças pulmonares utilizando imagens de raios-x do tórax / Mário Victor Gonzaga Monteiro. – 2024.
75 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2024.
Orientação: Prof. Dr. Fabrício Gonzalez Nogueira.
Coorientação: Prof. Dr. Victor Hugo Costa de Albuquerque.

1. Deep Learning. 2. CNN. 3. Radiografias. 4. Explicabilidade. 5. Grad-CAM++. I. Título.
CDD 621.3

MÁRIO VICTOR GONZAGA MONTEIRO

REDES NEURAIS CONVOLUCIONAIS EXPLICÁVEIS PARA DETECÇÃO DE DOENÇAS PULMONARES UTILIZANDO IMAGENS DE RAIOS-X DO TÓRAX

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Ceará como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof. Dr. Fabrício Gonzalez Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Victor Hugo Costa de Albuquerque
Universidade Federal do Ceará (UFC)

_____
Eng. MSc. Bruno Luiz Faustino
Universidade Federal do Ceará (UFC)

Ao meu pai, com quem pouco pude compartilhar lembranças, mas que nunca deixou de estar em meus pensamentos.

AGRADECIMENTOS

A todos os professores e ensinadores que tive e ainda tenho comigo. Os quais definitivamente muito me ajudaram a ser quem sou hoje.

Ao ELOS TEC, pelo apoio e prontidão em me ajudar neste projeto.

Ao Prof. Dr. Fabrício Gonzalez Nogueira, por sua total solicitude e atenção como meu orientador.

Ao Prof. Dr. Victor Hugo Costa de Albuquerque, que sempre se mostrou acessível aos meus pedidos e ideias, me norteando durante este trabalho.

À Universidade Federal do Ceará e às amizades construídas durante meu tempo como aluno, que carrego com carinho e apreço.

À École Centrale de Lille e às amizades que pude fazer enquanto aluno da mesma, onde pude expandir mais ainda meu conhecimento e minhas experiências de vida.

Ao Programa de Educação Tutorial (PET), uma das minhas maiores famílias, que me formou como ser humano. Em especial, ao Prof. Dr. René Pastor Torrico Bascopé, tutor do projeto durante meus quase três anos no PET.

Aos meus amigos, seja que conheci na UFC, na ECL ou fora do ambiente acadêmico. Sem dúvidas sou formado e lapidado por cada um de vocês.

Ao Miguel, meu companheiro de casa durante meus dois anos na França, que se tornou como um irmão mais velho para mim.

À família da minha namorada, que sempre me apoiou e tratou como parte da família.

À minha namorada e companheira de curso, por sempre me apoiar acadêmica e humanamente.

À minha família, que mesmo distante em boa parte da minha vida, nunca deixou de se preocupar e de manter apreço para comigo.

À minha irmã, cuja presença foi essencial durante minha infância.

À minha mãe, que também desempenhou o papel de meu pai, sendo a principal razão de quem eu sou hoje.

“O homem não pode refazer-se sem sofrer, afinal, ele é tanto o mármore quanto o escultor.”

(Alexis Carrel)

RESUMO

Este trabalho apresenta o desenvolvimento de um modelo de *Deep Learning* para o diagnóstico automatizado de múltiplas patologias em radiografias de tórax, utilizando técnicas de explicabilidade como o Grad-CAM++ e o LIME. A proposta visa não apenas identificar doenças em imagens médicas, mas também oferecer uma explicação visual que ajude a validar as previsões do modelo, promovendo uma maior confiança na sua utilização em ambientes clínicos. Para isso, foi utilizado o NIH *Chest X-ray Dataset*, uma das maiores bases de dados públicas de radiografias de tórax, contendo mais de 100.000 imagens rotuladas com múltiplas patologias. O dataset foi reduzido e balanceado para garantir uma distribuição mais equitativa das classes de doenças e uma melhor representatividade no treinamento do modelo. Foi empregada a arquitetura DenseNet-121 com *Transfer Learning*, escolhida por sua eficiência em tarefas de classificação de imagens e por ser amplamente utilizada na literatura médica para o diagnóstico de patologias. Além disso, técnicas como o *dropout* foram aplicadas para reduzir o risco de *overfitting*, e o modelo foi avaliado utilizando métricas de desempenho padrão, como o AUC-ROC e o *Hamming Loss*. O processo de avaliação incluiu uma etapa importante de explicabilidade, onde utilizamos o Grad-CAM++ para gerar mapas de calor das regiões da imagem que mais contribuíram para o diagnóstico, e o LIME para identificar os atributos mais relevantes na decisão do modelo. Essa abordagem foi validada por profissionais de saúde, que avaliaram as previsões e as explicações visuais geradas, fornecendo feedback qualitativo e quantitativo. A análise dos resultados demonstrou que o modelo apresentou boas performances em doenças como Cardiomegalia e Edema, com valores de AUC-ROC acima de 0.85. Doenças mais difíceis de identificar, como Infiltração, apresentaram valores mais modestos, refletindo a complexidade inerente ao diagnóstico dessas condições. Os resultados indicam que o uso de modelos de *Deep Learning,* aliado a técnicas de explicabilidade, pode ser uma ferramenta valiosa na prática clínica, especialmente ao proporcionar maior transparência nas previsões, facilitando a adoção dessas tecnologias pelos profissionais de saúde. No entanto, há limitações, como o baixo número de especialistas envolvidos na avaliação e as dificuldades em identificar corretamente múltiplas doenças concomitantes. Trabalhos futuros poderão incluir uma avaliação mais ampla com outros especialistas, além de explorar novas arquiteturas de rede e técnicas de explicabilidade mais robustas.

Palavras-chave: *Deep Learning*. Explicabilidade. CNN. Grad-CAM++. Radiografias.

ABSTRACT

This work presents the development of a Deep Learning model for the automated diagnosis of multiple pathologies in chest X-rays, using explainability techniques such as Grad-CAM++ and LIME. The goal is not only to identify diseases in medical images but also to provide a visual explanation that helps validate the model's predictions, fostering greater confidence in its use in clinical environments. The study employed the NIH Chest X-ray Dataset, one of the largest publicly available chest X-ray databases, containing over 100,000 images labeled with multiple pathologies. The dataset was reduced and balanced to ensure a more equitable distribution of disease classes and better representativity during model training. The DenseNet-121 architecture with transfer learning was utilized, chosen for its efficiency in image classification tasks and its extensive use in the medical literature for diagnosing pathologies. Additionally, techniques like dropout were applied to reduce overfitting, and the model was evaluated using standard performance metrics such as AUC-ROC and Hamming Loss. The validation process included an essential explainability step, where we used Grad-CAM++ to generate heatmaps highlighting the regions of the image that contributed the most to the diagnosis, and LIME to identify the most relevant features in the model's decision-making process. This approach was validated by healthcare professionals who evaluated the model's predictions and the visual explanations generated, providing both qualitative and quantitative feedback. The analysis of the results demonstrated that the model showed good performance in diseases such as Cardiomegaly and Edema, with AUC-ROC values above 0.85. More challenging conditions, like Infiltration, presented more modest values, reflecting the inherent complexity of diagnosing these diseases. The results suggest that the use of Deep Learning models, combined with explainability techniques, can be a valuable tool in clinical practice, especially by providing greater transparency in predictions, thus facilitating the adoption of these technologies by healthcare professionals. However, there are limitations, such as the low number of specialists involved in the validation and the challenges in accurately identifying multiple concomitant diseases. Future work could include broader validation with additional specialists, as well as exploring new network architectures and more robust explainability techniques.

Keywords: Deep Learning. Explainability. CNN. Grad-CAM++. X-rays.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA - Inteligência Artificial

XAI – Inteligência Artificial Explicável

CNN - Redes Neurais Convolucionais

Grad-CAM++ - *Gradient-Weighted Class Activation Mapping++*

LIME - *Local Interpretable Model-agnostic Explanations*

AUC-ROC - *Receiver Operating Characteristic Area Under the Curve*

ReLU - *Rectified Linear Unit*

NIH - *National Institutes of Health*

NLP - Processamento de Linguagem Natural

E/S - Entrada/Saída

DenseNet - *Densely Connected Convolutional Networks*

Atel - *Atelectasis*

Card - *Cardiomegaly*

Effu - *Effusion*

Infi - *Infiltration*

Nodu - *Nodule*

Pne1 - Pneumonia

Pne2 - *Pneumothorax*

Cons - *Consolidation*

Edem - Edema

Emph - *Emphysema*

Fibr - *Fibrosis*

P.T. – *Pleural Thickening*

Hern - *Hernia*

ViTs - *Transformers* Visuais

GPU – *Graphics Processing Unit*

FP16 – *Half-precision floating-point*

FP32 – *Single-precision floating point*

SUMÁRIO

1. INTRODUÇÃO

Nas últimas décadas, os avanços no campo da inteligência artificial (IA), em particular o *Deep Learning*, transformaram radicalmente a análise de imagens médicas. As Redes Neurais Convolucionais (CNNs), inicialmente propostas para reconhecimento de padrões visuais, estabeleceram um princípio fundamental que seria adotado na área médica para análise de imagens como raios-X, ressonâncias magnéticas e tomografias computadorizadas (LeCun *et al*., 1998).

Mais recentemente, a aplicação de CNNs na detecção de pneumonia em radiografias de tórax, utilizando grandes conjuntos de dados trouxe importantes contribuições para o campo (Rajpurkar *et al*., 2017). Relatos indicam que os modelos alcançaram desempenho comparável ou superior ao de radiologistas em certas condições, reforçando a capacidade do aprendizado profundo em diagnósticos médicos complexos. Além disso, a expansão dessas técnicas para incluir incertezas nos rótulos das patologias, como demonstrado com o conjunto de dados *CheXpert*, tornou o processo de rotulagem mais realista e aplicável à prática clínica (Irvin *et al*., 2019).

Entretanto, apesar dos avanços na precisão e generalização dos modelos, a chamada "caixa-preta" das redes neurais continua sendo uma barreira significativa para sua adoção ampla em ambientes clínicos. Estudos pioneiros em explicabilidade no *Deep Learning* destacam a necessidade de tornar os modelos mais interpretáveis para os usuários finais, principalmente em áreas sensíveis como a medicina (Zhang *et al*., 2018). A transparência nas decisões dos modelos é crucial para a confiança dos profissionais de saúde, que precisam entender como e por que o modelo chegou a uma determinada conclusão.

Como resultado, técnicas de explicabilidade, como o Grad-CAM (*Gradient-Weighted Class Activation Mapping*) e sua evolução Grad-CAM++, emergiram como soluções promissoras para fornecer insights visuais sobre as decisões do modelo (Selvaraju *et al*., 2017; Chattopadhyay *et al*., 2018). Pesquisas demonstram que o Grad-CAM++ melhora a precisão em cenários onde múltiplos objetos ou características estão presentes na imagem, tornando-se ideal para casos médicos com sobreposição de patologias (Chattopadhyay *et al*., 2018).

Além disso, revisões sobre o impacto e a aplicabilidade das técnicas de explicabilidade em IA ressaltam que a adoção dessas abordagens está diretamente relacionada à capacidade de especialistas confiarem nas previsões dos modelos, especialmente em áreas críticas como a medicina (Samek *et al*., 2017).

A aplicação dessas técnicas em diagnósticos médicos tem mostrado resultados promissores. Revisões sobre o uso de aprendizado profundo em imagens médicas apontam a explicabilidade como uma das principais áreas de pesquisa para melhorar a aceitação dessas tecnologias pelos profissionais de saúde (Lundervold e Lundervold, 2019). Além disso, estudos fornecem uma visão abrangente sobre os desafios e as melhores práticas na criação de modelos explicáveis em medicina (Tjoa e Guan, 2020).

1.1. Justificativa

Este trabalho se justifica pela urgência em desenvolver modelos de *Deep Learning* que não sejam apenas precisos, mas também transparentes e interpretáveis para os profissionais da saúde. O uso de técnicas como o Grad-CAM++ não só melhora a confiança dos médicos nas previsões automáticas, mas também permite que eles validem clinicamente as decisões, assegurando que os modelos identifiquem corretamente as áreas mais relevantes das imagens médicas. Ao focar em patologias importantes, como Cardiomegalia, Pneumonia e Edema, este estudo busca fornecer contribuições tanto na melhoria do desempenho dos modelos quanto na confiança de sua adoção na prática clínica, oferecendo suporte visual que torna o diagnóstico mais acessível e verificável pelos especialistas.

Adicionalmente, a literatura corrente sugere que há uma lacuna entre a performance dos modelos e a aceitação por parte dos profissionais médicos. Argumenta-se que a aplicabilidade clínica de modelos de IA será muito mais aceitável se forem acompanhados de explicações que forneçam suporte na tomada de decisão, especialmente em condições críticas como doenças cardíacas e pulmonares (Kim *et al*., 2020). Portanto, este projeto almeja preencher essa lacuna, validando o modelo com profissionais da saúde e propondo uma metodologia de avaliação qualitativa e quantitativa para garantir que as previsões estejam alinhadas com as expectativas clínicas.

1.2. Objetivos

O objetivo deste trabalho é desenvolver um modelo de *Deep Learning* capaz de realizar o diagnóstico automático de patologias em radiografias de tórax, utilizando técnicas de explicabilidade como Grad-CAM++ (*Gradient-weighted Class Activation Mapping*++) e LIME (*Local Interpretable Model-agnostic Explanations*) para gerar explicações visuais das previsões, promovendo maior confiança na utilização desses modelos em ambiente clínico.

Objetivos Específicos:

- Coletar e preparar um dataset robusto de radiografias de tórax com rótulos de múltiplas doenças, garantindo um conjunto de dados balanceado e relevante para o treinamento do modelo;
- Treinar e avaliar um modelo de rede neural convolucional, utilizando a arquitetura DenseNet para o diagnóstico de múltiplas patologias, incluindo Cardiomegalia, Pneumonia e Edema;
- Implementar as técnicas Grad-CAM++ e LIME, a fim de gerar explicações visuais que ajudem a interpretar as previsões do modelo, permitindo a visualização das áreas da imagem que mais contribuíram para a decisão. O Grad-CAM++ será usado para explicações globais com foco em regiões importantes na imagem, enquanto o LIME proporcionará explicações locais baseadas nas características individuais das previsões;
- Validar o modelo com profissionais da saúde, por meio de um questionário estruturado que analise a relevância das previsões e das explicações visuais fornecidas, buscando feedback qualitativo e quantitativo;
- Comparar os resultados com trabalhos anteriores da literatura, avaliando se o modelo proposto apresenta melhorias em termos de precisão e interpretabilidade, com base nas métricas padrões, como AUC-ROC (*Receiver Operating Characteristic Area Under the Curve*), *Hamming Loss*, e métricas qualitativas fornecidas pelos profissionais.

1.3. Organização do Trabalho

Este trabalho está dividido em cinco capítulos, conforme descrito a seguir:

- Capítulo 1 - Introdução: Apresenta a contextualização do problema, a justificativa do estudo, seus objetivos (geral e específicos) e a organização do Trabalho;
- Capítulo 2 - Fundamentação Teórica: Fornece uma revisão de literatura sobre os principais conceitos envolvidos no estudo, como Redes Neurais Convolucionais, *Deep Learning* aplicado à área médica, e técnicas de explicabilidade como Grad-CAM++;
- Capítulo 3 - Metodologia: Detalha os procedimentos adotados para a coleta e preparação dos dados, as ferramentas e tecnologias utilizadas, a arquitetura do modelo proposto, assim como o processo de treinamento e avaliação do modelo;
- Capítulo 4 - Resultados e Discussão: Apresenta os resultados obtidos durante os experimentos, com uma análise quantitativa das métricas de desempenho do modelo, além de discutir a avaliação das previsões realizada pelos profissionais de saúde;
- Capítulo 5 - Conclusão e Trabalhos Futuros: Finaliza o trabalho discutindo as principais conclusões, as limitações encontradas e as possíveis melhorias e desdobramentos futuros do projeto.

2. FUNDAMENTAÇÃO TEÓRICA

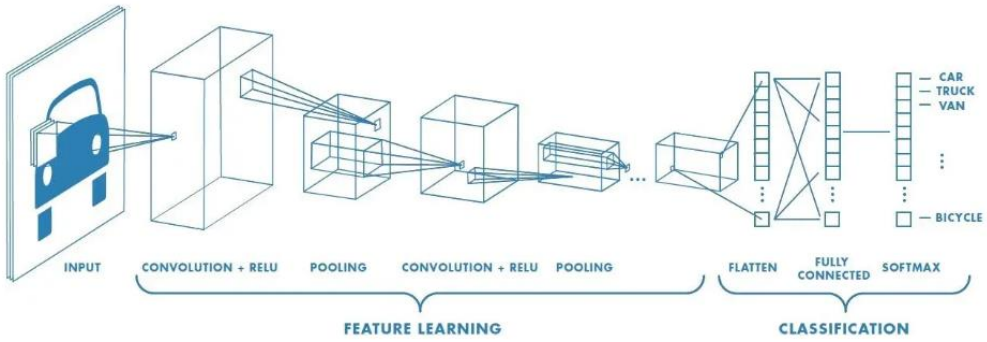
Nas últimas décadas, os avanços no campo da inteligência artificial, em particular o *Deep Learning*, transformaram radicalmente a análise de imagens médicas. As Redes Neurais Convolucionais, inicialmente propostas para reconhecimento de padrões visuais, estabeleceram um princípio fundamental que seria adotado na área médica para análise de imagens como raios-X, ressonâncias magnéticas e tomografias computadorizadas (LeCun *et al*., 1998).

2.1. Redes Neurais Convolucionais (CNNs)

As Redes Neurais Convolucionais surgiram como uma das principais arquiteturas de Deep Learning para tarefas de classificação e análise de imagens.

Uma CNN é composta por várias camadas especializadas que permitem a extração de características relevantes da imagem. Ao contrário das redes neurais tradicionais, que tratam os dados de forma tabular, as CNNs preservam as relações espaciais entre os pixels da imagem. Isso é fundamental para reconhecer estruturas visuais complexas, como bordas, texturas e objetos inteiros.

Figura 1 – Exemplo de CNN e suas camadas



Fonte: Adaptado de *Saturn Cloud* (2018).

As camadas mais comuns em uma CNN são:

2.1.1. Camadas Convolucionais

A camada convolucional é o núcleo das CNNs. Seu objetivo é aplicar um filtro convolucional (ou *kernel*) sobre a imagem de entrada, gerando mapas de ativação que destacam características visuais importantes, como bordas, cantos e texturas. O processo de convolução é matematicamente expresso como:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n)$$

Onde:

- $S(i,j)$ é o resultado da convolução no ponto i,j,
- $I(i+m, j+n)$ representa o valor do pixel na posição $(i+m, j+n)$,
- $K(m,n)$ é o filtro convolucional aplicado,
- Os somatórios percorrem as dimensões do kernel m,n,
- $*$ denota a operação de convolução.

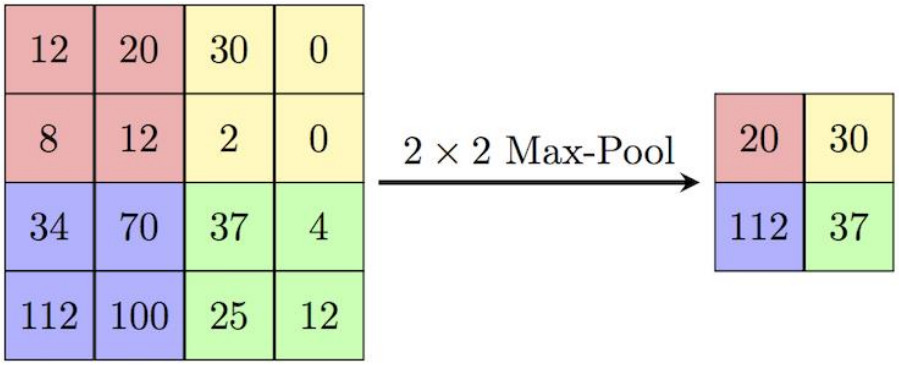
Cada filtro convolucional é movido através da imagem, produzindo um mapa de ativação que realça características específicas. Quanto mais filtros são aplicados, mais recursos a rede pode capturar. Cada filtro pode aprender a detectar uma característica diferente, como linhas horizontais, bordas verticais ou até formas complexas em camadas mais profundas.

2.1.2. Camadas de *Pooling*

Depois da convolução, normalmente aplica-se uma camada de *Pooling*, cuja função é reduzir a dimensionalidade dos mapas de ativação, mantendo as informações mais importantes. Isso reduz o custo computacional e ajuda a evitar o *overfitting,* pois diminui a quantidade de parâmetros.

Esse processo reduz a resolução espacial da imagem, mas mantém as características mais salientes, facilitando a generalização do modelo. O método mais utilizado é o *Max Pooling*, onde somente o maior valor de certa região é escolhido e mantido.

Figura 2 – Exemplo de *Max Pooling*



Fonte: Adaptado de *Computer Science Wiki* (2018).

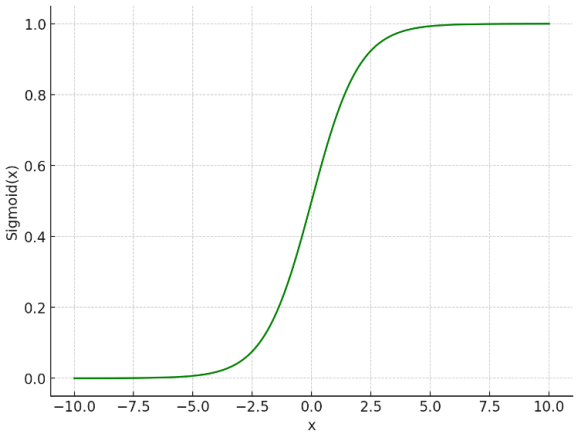
2.1.3. Funções de Ativação

As funções de ativação são aplicadas após a operação de convolução para introduzir não-linearidade no modelo, permitindo que a rede aprenda representações mais complexas. Por exemplo, a função sigmoide é definida como:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Ela transforma qualquer valor de entrada em uma saída entre 0 e 1. Isso é particularmente útil em problemas de classificação binária, onde o resultado pode ser interpretado como uma probabilidade. Por conta dessa característica, a função sigmoide é frequentemente utilizada em camadas finais de redes neurais que realizam classificação binária.

Figura 3 – Função sigmoide



Fonte: O autor.

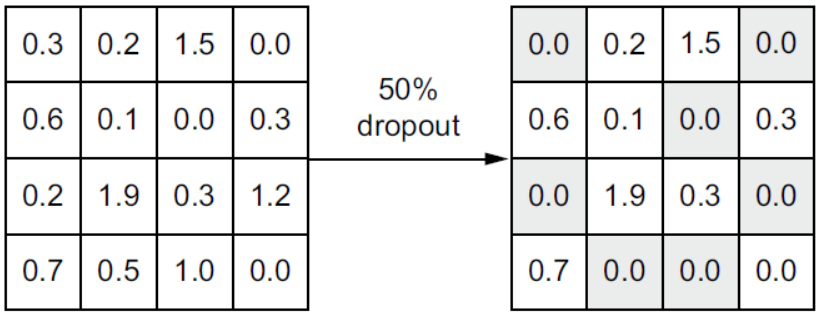
2.1.4. Regularização

Para evitar *overfitting*, ou seja, o sobreajuste dos dados, as CNNs geralmente incluem técnicas de regularização, sendo as mais comuns:

- *Dropout*: Desativa aleatoriamente uma fração de neurônios durante o treinamento, o que impede que a rede se torne muito dependente de um conjunto específico de neurônios.
- *Batch Normalization*: Normaliza as ativações de cada camada para que tenham média zero e variância unitária, acelerando o treinamento e estabilizando a rede.

A regularização é crucial para melhorar a generalização do modelo em novos dados, garantindo que ele não memorize o dataset de treino.

Figura 4 – Aplicação de *dropout* na rede



Fonte: Adaptado de *CHOLLET (2021)*

2.2. Inteligência Artificial Explicável (XAI)

Embora os modelos de *Deep Learning*, especialmente as CNNs, tenham alcançado resultados extraordinários em diversas tarefas, eles ainda sofrem de um problema fundamental: sua natureza de "caixa-preta". Diferentemente de modelos mais simples, como regressão linear ou árvores de decisão, onde as relações entre os dados e as previsões são claras e interpretáveis, as redes neurais envolvem uma complexa rede de pesos e ativações, dificultando a compreensão de como e por que o modelo chegou a uma determinada decisão.

Essa falta de interpretabilidade torna-se especialmente problemática em áreas críticas como a medicina, onde a transparência nas decisões é crucial. Profissionais de saúde precisam entender as razões por trás das previsões feitas por um modelo de IA antes de confiar em suas sugestões. Isso é particularmente importante no contexto do diagnóstico médico, onde erros podem ter consequências graves.

Diversas técnicas de explicabilidade foram desenvolvidas para mitigar esse problema, buscando fornecer uma compreensão mais clara das decisões tomadas pelos modelos de *Deep Learning*. Essas técnicas podem ser divididas em dois tipos principais:

- Explicabilidade Global: Fornece uma visão ampla de como o modelo toma suas decisões de forma geral, aplicando-se a todas as entradas possíveis.
- Explicabilidade Local: Foca em uma instância específica de entrada, explicando por que o modelo tomou uma decisão particular para aquela entrada específica.

Duas das técnicas mais emergentes e promissoras para explicabilidade em modelos de *Deep Learning* foram utilizadas neste trabalho: o Grad-CAM++ e o LIME. Ambas foram projetadas para oferecer maior transparência em redes neurais complexas, mas com abordagens diferentes.

2.2.1. Grad-CAM++

O Grad-CAM++ é uma técnica de explicabilidade visual que gera mapas de calor para indicar as regiões de uma imagem que mais contribuíram para a decisão de um modelo de *Deep Learning*. O Grad-CAM++ é uma extensão do Grad-CAM, projetado para lidar melhor com situações em que múltiplos objetos estão presentes na imagem ou quando a decisão do modelo é baseada em múltiplas regiões da imagem.

Esta técnica é particularmente útil em redes neurais convolucionais, pois permite a interpretação das previsões sem a necessidade de modificar a arquitetura do modelo ou retreinar o modelo para obter explicações.

O Grad-CAM++ gera mapas de ativação ponderados que indicam quais regiões da imagem foram mais importantes para uma determinada previsão do modelo. A técnica utiliza os gradientes das ativações da última camada convolucional em relação à saída de interesse (classe-alvo) para ponderar a importância de cada neurônio nas ativações da camada convolucional.

Em uma rede neural convolucional, as camadas convolucionais produzem mapas de ativação A^k, que contêm as respostas de cada filtro aplicado à imagem de entrada. O Grad-CAM++ utiliza esses mapas de ativação para identificar quais partes da imagem são mais relevantes para a previsão.

Seja A^k o mapa de ativação da k-ésima camada convolucional, onde k representa o número de filtros da camada. Cada filtro responde a diferentes características da imagem, como bordas, texturas ou formas específicas.

O Grad-CAM++ melhora o Grad-CAM ao introduzir um cálculo mais refinado dos pesos que são aplicados aos mapas de ativação. Em vez de apenas utilizar o gradiente bruto, o Grad-CAM++ utiliza três derivadas para calcular as contribuições individuais de cada neurônio no mapa de ativação. Os pesos α_k^{ij} que representam a importância de cada pixel no mapa de ativação para a classe c são calculados da seguinte forma:

$$\alpha_k^{ij} = \frac{\frac{\partial^2 y^c}{\partial A_{ij}^k}}{2\frac{\partial^2 y^c}{\partial A_{ij}^k} + \sum_{m,n} A_{mn}^k \frac{\partial^3 y^c}{\partial A_{mn}^k}}$$

Essa equação refina a ponderação de cada neurônio, resultando em explicações mais detalhadas e precisas, especialmente em casos onde a decisão do modelo é baseada em várias partes da imagem.

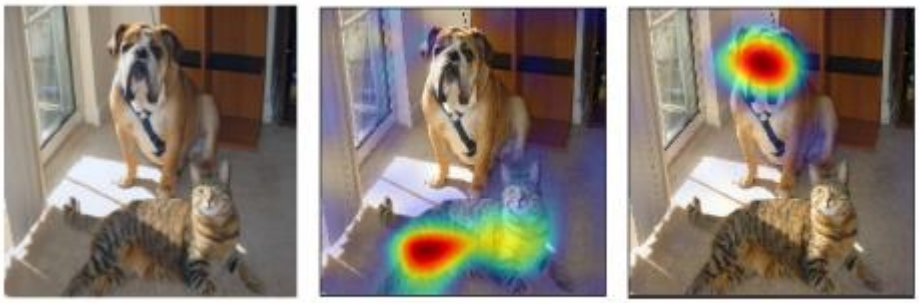
Com os pesos calculados, o Grad-CAM++ gera o mapa de calor somando os mapas de ativação ponderados pelos pesos correspondentes:

$$L^c_{\text{Grad-CAM++}} = ReLU(\sum_k \alpha_k A^k)$$

O operador ReLU (*Rectified Linear Unit*) é utilizado para garantir que apenas as ativações que contribuíram positivamente para a previsão da classe sejam consideradas no mapa final. O resultado final do Grad-CAM++ é um mapa de calor que pode ser sobreposto à imagem original para mostrar as regiões mais importantes. As áreas destacadas em vermelho indicam as regiões que tiveram maior influência na decisão do modelo.

Este mapa é particularmente útil para aplicações médicas, pois permite que os médicos visualizem se o modelo está focando nas áreas corretas da imagem ao fazer um diagnóstico, proporcionando uma explicabilidade mais robusta para a tomada de decisão.

Figura 5 – Mapa de calor produzido pelo Grad-CAM



Fonte: Adaptado de *DRAELOS* (2020).

2.2.2. LIME

O LIME, proposto por Ribeiro *et al*. (2016), é uma técnica que visa explicar as previsões de qualquer modelo de aprendizado de máquina, fornecendo explicações locais e interpretáveis para instâncias individuais. A ideia central do LIME é ajustar um modelo simples e interpretável, para aproximar o comportamento do modelo mais complexo em torno de uma entrada específica. Esse modelo local, mais simples, pode então ser usado para fornecer uma explicação sobre como o modelo complexo tomou sua decisão.

Esta técnica é especialmente útil em cenários médicos, como o diagnóstico a partir de imagens de radiografias, pois permite que os profissionais de saúde compreendam por que o modelo está classificando uma determinada imagem como patológica ou não, mostrando quais partes da imagem (ou características) são mais relevantes para essa decisão.

O LIME funciona da seguinte maneira:

A primeira etapa é criar versões perturbadas da instância original de entrada. Essas perturbações são pequenas modificações que podem incluir remover ou alterar partes da imagem. Isso gera um conjunto de novas instâncias ligeiramente diferentes da original.

Em seguida, essas instâncias perturbadas são enviadas para o modelo original para obter as previsões. O objetivo é observar como pequenas mudanças na entrada afetam a previsão do modelo.

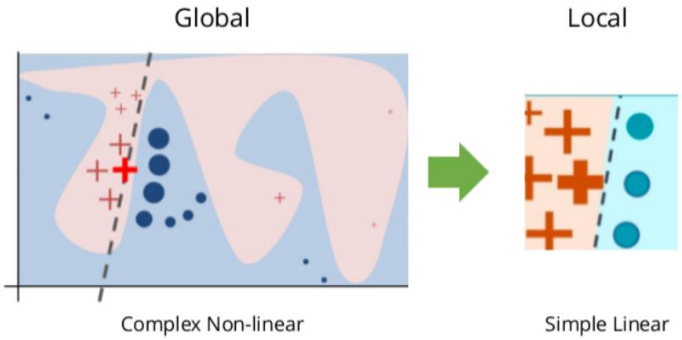
O LIME ajusta um modelo simples (geralmente uma regressão linear) para aproximar as previsões do modelo original, mas apenas no entorno da instância original perturbada. Isso permite que a explicação seja local, concentrando-se na vizinhança da instância original, sem tentar generalizar para todo o espaço de entrada.

Por fim, o LIME atribui pesos às diferentes características da entrada original, indicando quais delas foram mais importantes para a previsão do modelo. Essas características podem ser interpretadas facilmente, pois o modelo local é simples e transparente.

Quando aplicado a uma radiografia de tórax, o LIME pode destacar quais regiões (superpixels) foram mais importantes para a previsão de uma determinada doença. Suponhamos que o modelo preveja Cardiomegalia em uma radiografia. O LIME pode destacar os superpixels da imagem onde o modelo focou mais, mostrando, por exemplo, a área do coração como a mais relevante para a previsão.

Ao combinar o LIME com o Grad-CAM++, podemos oferecer uma explicabilidade robusta, tanto no nível global quanto local, aumentando a confiança no uso dessas tecnologias em ambientes clínicos.

Figura 6 – Mudança de granularidade para a explicabilidade via LIME



Fonte: Adaptado de *C3 AI* (2024).

2.3. Dataset NIH Chest X-ray

O *NIH Chest X-ray Dataset*, também conhecido como *ChestX-ray14*, é um dos maiores *datasets* públicos de imagens de radiografias de tórax disponível para a pesquisa em diagnóstico médico assistido por inteligência artificial. O *dataset* foi lançado em 2017 pelo *National Institutes of Health* (NIH) com o objetivo de fomentar o desenvolvimento de modelos de aprendizado de máquina capazes de realizar diagnósticos automatizados de várias condições patológicas presentes em radiografias de tórax.

O *ChestX-ray14* contém um total de 112.120 imagens de radiografias de tórax de 30.805 pacientes. Essas imagens foram rotuladas com 14 diferentes condições médicas, tornando-o um *dataset* extremamente valioso para o desenvolvimento de modelos de diagnóstico multi-patológico. Cada imagem pode conter mais de uma condição rotulada, permitindo que os pesquisadores desenvolvam modelos de classificação *multilabel*.

As 14 condições patológicas incluídas no dataset são:

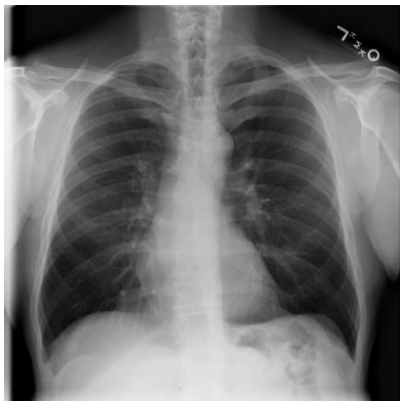
- *Atelectasis* (Atelectasia)
- *Cardiomegaly* (Cardiomegalia)
- *Effusion* (Derrame Pleural)
- *Infiltration* (Infiltração)
- *Mass* (Massa)
- *Nodule* (Nódulo)
- Pneumonia
- *Pneumothorax* (Pneumotórax)

- *Consolidation* (Consolidação Pulmonar)
- *Edema* (Edema Pulmonar)
- *Emphysema* (Enfisema)
- *Fibrosis* (Fibrose Pulmonar)
- *Pleural Thickening* (Espessamento Pleural)
- *Hernia* (Hérnia)

Além disso, o *dataset* também inclui informações demográficas dos pacientes, como sexo e idade, além de dados sobre os procedimentos radiológicos realizados, permitindo estudos mais amplos que envolvem a correlação entre variáveis clínicas e imagens.

O *NIH Chest X-ray Dataset* contém imagens em escala de cinza com resolução variada, que representam radiografias de tórax anteriores (*frontal view*). Junto com as imagens, o *dataset* inclui um arquivo CSV contendo informações de rótulos para cada uma das 112.120 imagens, detalhando as patologias associadas a cada radiografia.

Figura 7 – Amostra de raio-x torácico do *dataset* NIH



Fonte: *National Institutes of Health Clinical Center*.

Esse formato torna o *dataset* acessível para a aplicação direta de modelos de *Deep Learning*, uma vez que os rótulos já estão organizados de forma clara e associada a cada imagem. O *dataset* abrange uma grande diversidade de condições patológicas. Isso permite que pesquisadores desenvolvam modelos *multilabel*, que são capazes de diagnosticar múltiplas condições em uma única imagem, o que simula situações reais em que um paciente pode apresentar mais de uma patologia.

Apesar de seu valor inegável, a maioria dos rótulos no *dataset* foi gerada automaticamente a partir de relatórios médicos usando NLP (Processamento de Linguagem Natural), o que pode introduzir incertezas nos rótulos. Ademais, um dos maiores obstáculos para a adoção de IA no ambiente clínico é a falta de confiança dos profissionais em relação às decisões automáticas de um modelo. A implementação conjunta do Grad-CAM++ e do LIME visa reduzir essa barreira, fornecendo justificativas visuais que permitem aos médicos compreenderem melhor as decisões do modelo e, assim, aumentarem sua confiança nas previsões.

Além das tradicionais métricas quantitativas, este trabalho propõe uma avaliação qualitativa com especialistas, algo ainda pouco comum em trabalhos dessa natureza. A avaliação qualitativa não só testa a precisão das previsões, mas também avalia se as explicações geradas pelo modelo são intuitivas e úteis para médicos e radiologistas no ambiente clínico.

3. METODOLOGIA

A metodologia deste trabalho foi rigorosamente desenvolvida para garantir a obtenção de resultados confiáveis e comparáveis aos estudos mais avançados na área de diagnósticos médicos baseados em *Deep Learning*. O processo metodológico foi estruturado em etapas claras, abrangendo desde a extração e transformação dos dados até o treinamento e validação do modelo proposto. Todas as etapas foram executadas no ambiente do Google *Colaboratory Pro* (Colab Pro), utilizando uma GPU NVIDIA Tesla T4, visando otimizar a eficiência computacional e a escalabilidade do modelo.

3.1. Justificativa para o uso do Colab Pro e da GPU NVIDIA Tesla T4

O Google Colab Pro foi selecionado como ambiente de desenvolvimento devido à sua capacidade de fornecer recursos de computação em nuvem de alto desempenho, incluindo acesso a unidades de processamento gráfico (GPUs) poderosas, como a NVIDIA Tesla T4. A escolha da GPU T4 se justifica por sua arquitetura Turing e suporte ao processamento em precisão simples (FP32) e meia precisão (FP16), permitindo acelerar o treinamento de redes neurais profundas e otimizar o uso de memória (NVIDIA, 2018). Essas características são cruciais ao trabalhar com conjuntos de dados de grande escala, como o *NIH ChestX-ray14*.

A GPU T4 dispõe de 16 GB de memória VRAM, o que, embora insuficiente para carregar todo o *dataset* simultaneamente, permitiu, em conjunto com técnicas de processamento em lotes (*chunks*), o treinamento eficiente do modelo sem comprometer a integridade dos dados. Além disso, a T4 oferece suporte a recursos avançados, como treinamento distribuído e otimizações de memória, que são particularmente úteis ao treinar arquiteturas complexas como a DenseNet-50.

3.2. Extração e Transformação dos Dados

A etapa inicial do estudo envolveu a coleta, organização e transformação dos dados, assegurando a integridade e a homogeneidade necessárias para o treinamento e validação do modelo. Para garantir a comparabilidade com estudos anteriores, foram seguidas as divisões originais do conjunto de dados fornecido pelo NIH, bem como aplicadas técnicas de manipulação de dados para contornar limitações computacionais.

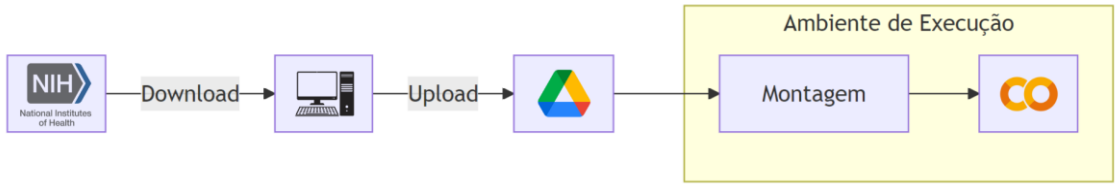
3.2.1. Coleta do Dataset

Os dados utilizados neste trabalho foram obtidos diretamente do site oficial do *National Institutes of Health*, que disponibiliza o *ChestX-ray14*, um conjunto composto por 112.120 imagens de radiografias de tórax rotuladas com 14 diferentes condições patológicas. O *download* foi realizado através de link público fornecido pelo NIH, garantindo que o *dataset* utilizado estivesse atualizado e em conformidade com os padrões éticos e legais de compartilhamento de dados médicos (NIH Clinical Center, 2017).

3.2.2. Transferência para o Ambiente de Execução

Após a obtenção do *dataset*, procedeu-se à transferência das imagens e dos arquivos de rótulos para o Google Drive, integrado ao ambiente do Google Colab Pro. Esta abordagem facilitou o acesso e a manipulação dos arquivos durante o processo de transformação e treinamento, aproveitando a conexão estável e a capacidade de armazenamento do Google Drive (Google, 2020).

Figura 8 – Fluxograma da extração dos dados



Fonte: O autor.

3.2.3. Pré-processamento dos Dados

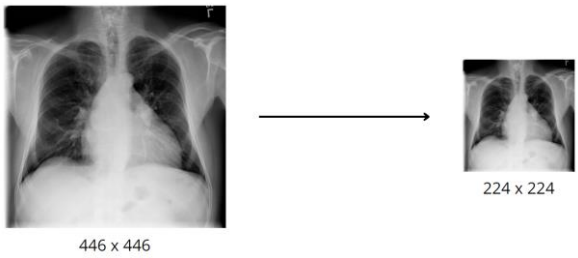
O pré-processamento dos dados foi fundamental para adaptar as imagens e os rótulos ao formato adequado para o treinamento do modelo. Este processo incluiu:

- As imagens foram associadas aos seus respectivos rótulos patológicos utilizando o arquivo CSV fornecido pelo NIH, denominado *Data_Entry_2017.csv*. Este arquivo contém o nome de cada imagem e as condições patológicas associadas. Cada imagem foi vinculada a um vetor binário *multilabel*, onde cada entrada indica a presença (1) ou ausência (0) de

uma das 14 condições patológicas. Esta abordagem permite modelar o problema como uma tarefa de classificação *multilabel*, refletindo a possibilidade de uma única imagem apresentar múltiplas patologias simultaneamente;

- Para assegurar a homogeneidade e permitir comparações diretas com outros trabalhos, foram utilizadas as listas *train_val_list.txt* e *test_list.txt*, fornecidas junto ao *dataset*. Estas listas contêm a divisão oficial estabelecida pelo NIH para treinamento/validação e teste. Ao seguir esta divisão, garantiu-se que o conjunto de teste utilizado é idêntico ao de outros estudos, facilitando uma comparação justa dos resultados.
- As imagens originais, de dimensões variadas, foram redimensionadas para uma resolução de 224x224 pixels, garantindo compatibilidade com a entrada da arquitetura DenseNet-50, que requer imagens em um tamanho padrão (Huang *et al*., 2017). O redimensionamento foi realizado utilizando interpolação bilinear, preservando ao máximo a qualidade visual das imagens.

Figura 9 – Redimensionamento das imagens



Fonte: O autor.

- A normalização foi realizada dividindo-se os valores dos pixels por 255, padronizando-os para o intervalo [0, 1]. Esta normalização é uma prática comum em visão computacional, pois facilita o treinamento, contribui para a estabilidade numérica do modelo e permite que os pesos da rede sejam atualizados de forma mais eficiente durante o processo de otimização (Goodfellow *et al*., 2016).

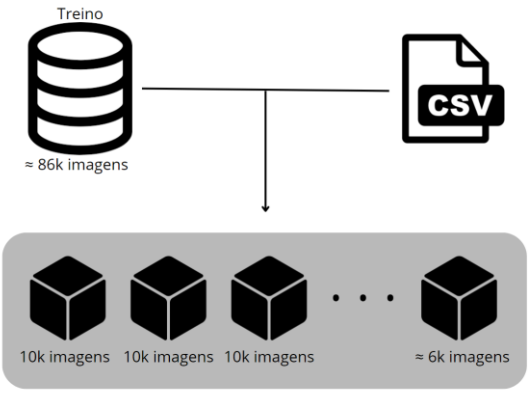
3.2.4. Processamento em Lotes (*Chunks*)

Devido às limitações de memória do ambiente, mesmo com o uso da GPU T4, foi necessário dividir o *dataset* em lotes menores para processar as imagens de forma gradual. O processamento em lotes permitiu evitar o consumo excessivo de memória e manteve o processamento eficiente.

O procedimento seguiu as seguintes etapas:

- As imagens foram divididas em lotes de 10.000 imagens. Este tamanho de lote foi escolhido empiricamente, balanceando a quantidade de dados processados por iteração e o uso de memória disponível. Este procedimento foi aplicado tanto ao conjunto de treino/validação quanto ao conjunto de teste.
- Para otimizar o tempo de leitura e escrita durante o treinamento, as imagens e os rótulos foram armazenados em arquivos no formato HDF5. Este formato é amplamente utilizado para armazenar grandes volumes de dados numéricos e oferece alto desempenho em operações de entrada/saída (E/S) (Folk *et al*., 2011). A utilização do HDF5 permitiu o acesso eficiente aos dados durante o treinamento, reduzindo a latência e melhorando o *throughput*.
- Em cada lote, as imagens foram armazenadas juntamente com seus rótulos, garantindo que cada *chunk* contivesse os dados completos necessários para o treinamento. Esta estratégia facilitou o carregamento eficiente dos dados durante o treinamento da rede neural, evitando a necessidade de buscas repetidas ou operações dispendiosas de leitura em disco.

Figura 10 – Representação do processamento em lotes



Fonte: O autor.

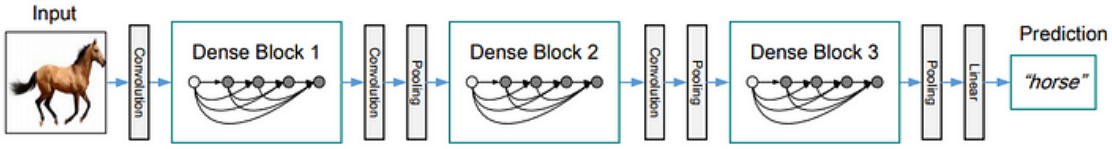
3.3. Arquitetura do Modelo

A DenseNet-50 (*Densely Connected Convolutional Networks*) foi a arquitetura escolhida para este trabalho devido à sua eficiência e capacidade de reutilização de características. A DenseNet foi proposta por Huang *et al*. (2017) e difere das CNNs tradicionais ao conectar cada camada diretamente a todas as camadas subsequentes, formando uma espécie de rede densa. Essa arquitetura tem várias vantagens sobre as redes convolucionais convencionais, incluindo:

- Reutilização de Características: Cada camada recebe como entrada todos os mapas de características das camadas anteriores, promovendo a reutilização de características extraídas anteriormente. Isso reduz a necessidade de aprender características redundantes e melhora a eficiência do modelo.
- Mitigação do Problema de Desvanecimento de Gradientes: As conexões diretas entre camadas facilitam o fluxo de gradientes durante o treinamento, permitindo que redes muito profundas sejam treinadas eficientemente sem sofrer com o desvanecimento ou explosão de gradientes.
- Eficiência de Parâmetros: Devido à reutilização extensiva de características, as DenseNets geralmente requerem menos parâmetros do que arquiteturas tradicionais, como a ResNet (He *et al*., 2016), reduzindo o risco de *overfitting* e otimizando o uso de recursos computacionais.

A versão DenseNet-50 usada neste trabalho tem 50 camadas, sendo uma escolha equilibrada entre profundidade e desempenho computacional. A versão DenseNet-121 poderia ter sido uma alternativa, mas o foco no equilíbrio entre tempo de processamento e eficiência motivou a escolha da versão com 50 camadas.

Figura 11 – Arquitetura geral das DenseNets



Fonte: Adaptado de *Louis Bouchard AI* (2021).

3.3.1. Transfer Learning

Para aprimorar o desempenho do modelo e acelerar o processo de treinamento, foi empregada a técnica de *Transfer Learning*. A DenseNet-50 foi inicializada com pesos pré-treinados no *dataset* ImageNet (Deng *et al*., 2009), que consiste em milhões de imagens distribuídas em milhares de categorias. O uso de pesos pré-treinados permite que o modelo aproveite características genéricas aprendidas anteriormente, como detecção de bordas e texturas, que são transferíveis para o domínio das imagens médicas (Yosinski *et al*., 2014).

Inicialmente, as camadas convolucionais da DenseNet-50 foram congeladas para preservar as características previamente aprendidas. Apenas as camadas finais, específicas para a tarefa atual, foram treinadas.

Esta abordagem otimiza o tempo de treinamento e melhora a performance, já que o modelo começa com uma base sólida de conhecimento visual.

3.3.2. Adaptação das Camadas de Saída e Função de Ativação

Considerando que o problema em questão é uma classificação *multilabel*, onde uma única imagem pode apresentar múltiplas patologias simultaneamente, a camada de saída da DenseNet-50 foi adaptada para conter 14 neurônios, correspondendo às 14 condições patológicas do *dataset*.

A função de ativação utilizada na camada de saída foi a sigmoide, adequada para tarefas *multilabel*, pois calcula independentemente a probabilidade de cada classe estar presente na imagem. A função sigmoide permite que o modelo atribua probabilidades independentes a cada patologia, refletindo a natureza não mutuamente exclusiva das classes.

3.3.3. Função de Perda

A função de perda escolhida foi a *Binary Crossentropy*, apropriada para problemas de classificação *multilabel* binária (Goodfellow *et al*., 2016). Esta função calcula a divergência entre as probabilidades previstas pelo modelo e os rótulos reais para

cada classe. A *Binary Crossentropy* é adequada para minimizar a discrepância entre as previsões do modelo e os rótulos verdadeiros em um contexto *multilabel*.

3.3.4. Métricas de Avaliação

A métrica primária utilizada para avaliar o desempenho do modelo foi a Área Sob a Curva Característica de Operação do Receptor (AUC-ROC) para cada classe individualmente. A AUC-ROC é amplamente utilizada em problemas de classificação médica devido à sua capacidade de medir a discriminação do modelo entre as classes positiva e negativa, independentemente do limiar de decisão (Fawcett, 2006).

A escolha da AUC-ROC se justifica especialmente em cenários com desbalanceamento de classes, como no *dataset* NIH *ChestX-ray14*, onde algumas patologias são raras e pouco estão presentes no conjunto de dados. Uma AUC próxima de 1 indica excelente capacidade do modelo em distinguir entre a presença e ausência de uma patologia.

Como métrica complementar, foi utilizada a *Hamming Loss*, que mede a proporção de rótulos incorretos em relação ao total de rótulos previstos, considerando todas as classes simultaneamente (Schapire & Singer, 2000). A *Hamming Loss* fornece uma medida do erro geral do modelo em predições *multilabel*, sendo útil para avaliar a performance em cenários onde múltiplas patologias podem coexistir.

3.3.5. *Dropout* e Validação Cruzada (*K-Fold Cross-Validation*)

Para prevenir o *overfitting* e melhorar a capacidade de generalização do modelo, foi aplicada a técnica de *Dropout*. Uma camada de *Dropout* com taxa de 0,5 foi inserida, desativando aleatoriamente 50% dos neurônios durante o treinamento. Esta estratégia reduz a coadaptação entre neurônios e força o modelo a aprender representações mais robustas e generalizáveis.

Para também assegurar a robustez e a generalização do modelo, foi utilizada a técnica de *K-Fold Cross-Validation*. O *dataset* de treinamento foi dividido em 5 subconjuntos estratificados, garantindo a representatividade de todas as classes em cada

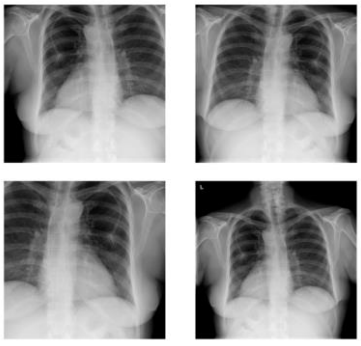
fold. O modelo foi treinado em 4 *folds* e validado no restante, repetindo o processo até que cada *fold* tenha servido como conjunto de validação uma vez (Kohavi, 1995). Este procedimento proporciona uma estimativa mais confiável do desempenho real do modelo, reduzindo a variabilidade associada a uma única divisão dos dados.

3.4. Treinamento do Modelo

O treinamento do modelo foi conduzido no Google Colab Pro, utilizando uma GPU NVIDIA Tesla T4, aproveitando sua capacidade computacional para acelerar o processo. As configurações adotadas foram:

- *Batch Size*: Um tamanho de lote de 32 foi selecionado, equilibrando a eficiência computacional e o uso de memória da GPU.
- Número de Épocas: O modelo foi treinado por 30 épocas, monitorando a perda e a métrica de validação para evitar *overfitting*.
- Otimizador: O otimizador Adam (Kingma & Ba, 2015) foi utilizado com uma taxa de aprendizado inicial de 1×10^{-4}. O Adam combina as vantagens dos otimizadores AdaGrad e RMSProp, adaptando as taxas de aprendizado para cada parâmetro e acelerando a convergência.
- *Data Augmentation*: Para aumentar a variabilidade dos dados e melhorar a generalização, técnicas de aumento de dados foram aplicadas, incluindo rotações aleatórias, deslocamentos horizontais e verticais, e inversão horizontal. Estas transformações simulam variações comuns nas radiografias e ajudam o modelo a ser mais robusto a mudanças na posição e orientação (Perez & Wang, 2017).

Figura 12 – Exemplo de *Data Augmentation*



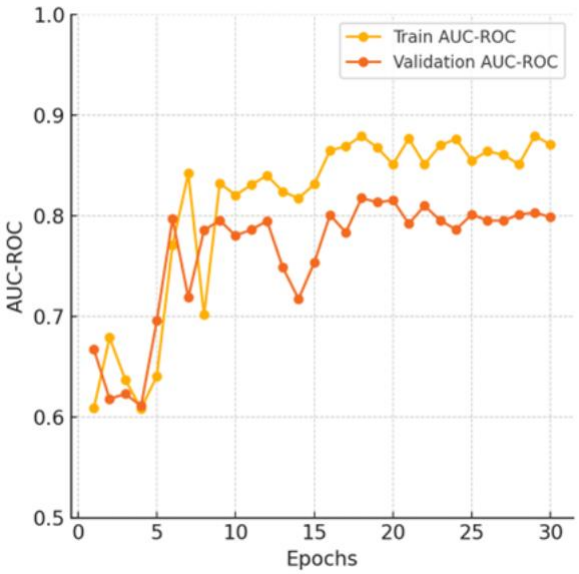
Fonte: O autor.

3.4.1. Tempo de Treinamento

O tempo total de treinamento variou de 6 a 8 horas para cada *fold* da validação cruzada, totalizando aproximadamente 37 horas de treinamento.

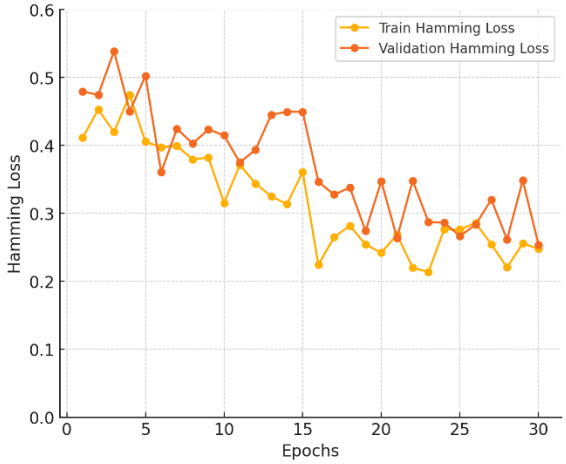
O uso do Google Colab Pro e da GPU NVIDIA Tesla T4 foi fundamental para viabilizar o treinamento em tempo hábil, aproveitando a aceleração por *hardware* e a infraestrutura de computação em nuvem.

Figura 13 – AUC-ROC de Treino e Validação



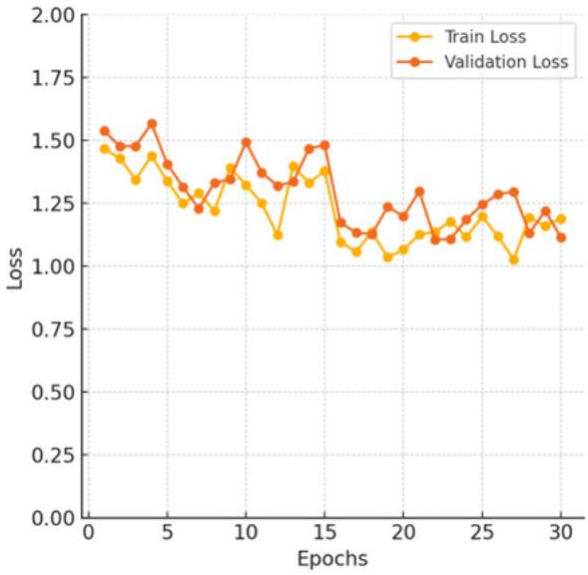
Fonte: O autor.

Figura 14 – *Hamming Loss* de Treino e Validação



Fonte: O autor.

Figura 15 – Perda de Treino e Validação



Fonte: O autor.

3.5. Aplicação de Técnicas de Explicabilidade: Grad-CAM++ e LIME

A interpretabilidade dos modelos de *Deep Learning* é fundamental, especialmente em contextos médicos, onde decisões baseadas em modelos precisam ser compreendidas e validadas por profissionais da saúde. Para este fim, foram aplicadas duas técnicas de explicabilidade: Grad-CAM++ e LIME. Essas técnicas permitem visualizar e interpretar as regiões das imagens que mais influenciaram as previsões do modelo, contribuindo para a transparência e confiança nas decisões automatizadas.

3.5.1. Grad-CAM++

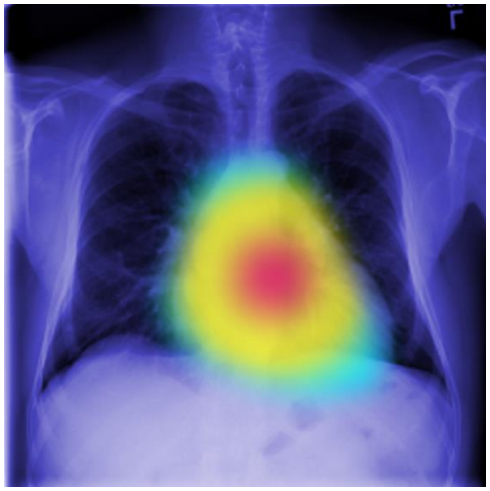
A técnica Grad-CAM++ foi empregada para gerar mapas de calor que destacam as regiões das radiografias de tórax que mais contribuíram para as previsões do modelo.

A implementação do Grad-CAM++ foi realizada utilizando a arquitetura DenseNet-50 treinada previamente. O Grad-CAM++ foi aplicado às saídas da última camada convolucional da rede, denominada *conv5_block3_out*.

Para cada imagem do conjunto de teste, o Grad-CAM++ seguiu os seguintes passos:

- Predição do Modelo: A imagem foi inserida no modelo DenseNet-50 para gerar uma previsão inicial das patologias presentes.
- Cálculo dos Gradientes: Foram calculados os gradientes das saídas correspondentes a cada patologia em relação às ativações da camada *conv5_block3_out*.
- Determinação dos Pesos de Importância: Utilizando os gradientes, foram computados os pesos que refletem a contribuição de cada ativação para a previsão final.
- Geração do Mapa de Ativação: Os pesos foram combinados com as ativações para produzir um mapa de ativação, que foi então redimensionado para corresponder às dimensões da imagem original.
- Superposição do Mapa de Calor: O mapa de ativação foi sobreposto à imagem original, permitindo visualizar as regiões que mais influenciaram a decisão do modelo.

Figura 16 – Grad-CAM++ aplicado



Fonte: O autor.

3.5.2. LIME

A técnica LIME foi empregada para fornecer explicações locais detalhadas para previsões específicas do modelo. Diferentemente do Grad-CAM++, que oferece uma visão geral das regiões de interesse, o LIME permite uma análise mais granular,

destacando quais partes da imagem foram mais influentes para a decisão em cada instância individual (Ribeiro, Singh & Guestrin, 2016).

Cada imagem selecionada foi dividida em superpixels, que são pequenas regiões de pixels com características semelhantes. Essa segmentação facilita a geração de perturbações localizadas na imagem. O LIME perturbou os superpixels substituindo-os por uma cor de fundo ou removendo-os completamente. Cada perturbação foi então inserida no modelo para observar como pequenas alterações na imagem afetavam a previsão.

Com base nas predições resultantes das perturbações, um modelo local simples foi ajustado para aproximar o comportamento do modelo complexo nas proximidades da imagem original. O LIME utilizou o modelo local para identificar quais superpixels tiveram maior impacto na previsão da patologia, destacando essas regiões como mais relevantes.

Figura 17 – LIME aplicado

Fonte: O autor.

3.6. Avaliação com profissionais da saúde

A avaliação com profissionais de saúde é uma parte crucial deste trabalho, pois garante que as previsões e explicações visuais geradas pelo modelo não são apenas estatisticamente robustas, mas também clinicamente relevantes e interpretáveis. Em

ambientes médicos, a confiança no uso de inteligência artificial vai além da precisão das previsões; é fundamental que os profissionais entendam como e por que o modelo tomou uma determinada decisão.

Dada a natureza crítica das decisões diagnósticas, especialmente em áreas como radiologia, a utilização de técnicas como Grad-CAM++ e LIME precisa ser validada quanto à sua utilidade prática. A principal motivação deste trabalho é garantir que o modelo seja explicável e verificável pelos especialistas, permitindo uma adoção mais segura e confiável de *Deep Learning* no ambiente clínico.

3.6.1. Processo de Avaliação

O processo de avaliação foi feito com base em imagens do conjunto de teste previamente processadas pelo modelo, para as quais foram gerados mapas de calor (Grad-CAM++) e explicações locais (LIME). Um conjunto selecionado de 6 imagens foi enviado para análise por cada profissional, abrangendo diferentes patologias. Das 6, 3 eram únicas e as outras 3 repetiam-se mais duas vezes para outros profissionais, visando a comparações entre diferentes avaliações.

Cada profissional recebeu as seguintes informações e ferramentas:

- Imagens originais da radiografia, sem nenhuma anotação.
- Imagens geradas pelos Grad-CAM++ e LIME indicando as regiões que o modelo considerou mais relevantes para a previsão.
- Questionário estruturado com 5 perguntas para avaliar a precisão e confiabilidade do modelo, bem como a relevância das explicações visuais.

3.6.2. Questionário de avaliação

O questionário consistiu em cinco perguntas chave, que foram formuladas para medir a aceitação das previsões do modelo e a confiança nas explicações visuais fornecidas. As perguntas e o processo de avaliação foram estruturados da seguinte forma:

Pergunta 1: Diagnóstico Visual (Imagem Original)

- Pergunta: "Com base na imagem, qual seria o seu diagnóstico para essa condição?"

Essa pergunta foi incluída para comparar o diagnóstico puramente visual feito pelos especialistas com o diagnóstico gerado pelo modelo. Isso permitiu avaliar a consistência entre a decisão dos especialistas e as previsões do modelo antes mesmo de considerar as explicações visuais geradas.

Pergunta 2: Relevância do Mapa de Calor (Imagem com Grad-CAM++)

- Pergunta: "O mapa de calor gerado pelo modelo destaca as áreas relevantes da imagem para o diagnóstico?"

Essa pergunta buscou verificar se o modelo, ao destacar regiões da imagem via Grad-CAM++, realmente focou em áreas clinicamente significativas para o diagnóstico da patologia. Se o especialista respondeu "Sim", significava que o modelo estava alinhado com a prática clínica; respostas como "Parcialmente" ou "Não" indicavam que as explicações visuais podem precisar de ajuste.

Pergunta 3: Precisão das Áreas Destacadas

- Pergunta: "Você concorda que as áreas destacadas pelo LIME são as mesmas que você usaria para fazer o diagnóstico?"

Essa pergunta foi um refinamento da anterior e visou garantir que as regiões exatas destacadas pelo modelo correspondessem às áreas que o especialista consideraria essenciais para o diagnóstico. Isso é crucial para garantir que o modelo esteja tomando decisões com base nas mesmas informações visuais que um humano usaria, aumentando a confiança no sistema.

Pergunta 4: Grau de Confiabilidade do Modelo

- Pergunta: "De 1 a 5, qual o nível de confiabilidade que você atribuiria ao modelo, com base no mapa de calor?"

Aqui, os especialistas puderam expressar quantitativamente o nível de confiança que tinham no modelo, com base nas explicações visuais fornecidas. Esse feedback quantitativo foi importante para medir a percepção geral de confiança dos especialistas no uso do Grad-CAM++ e do modelo em si.

Pergunta 5: Comentários Adicionais

- Pergunta: "Há alguma outra observação sobre a imagem ou sobre a interpretação feita pelo modelo?"

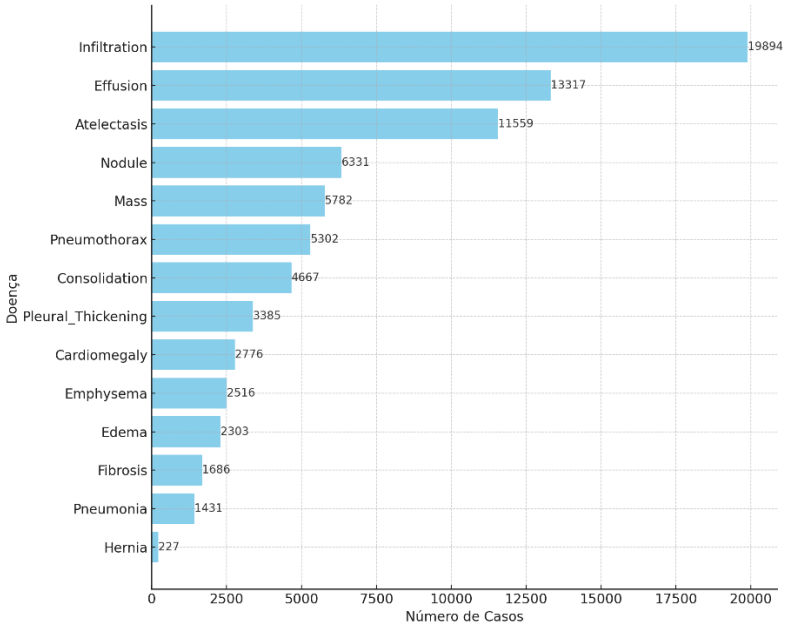
Essa pergunta aberta permitiu aos especialistas fornecerem comentários mais detalhados sobre os mapas de calor ou o processo de explicação. Isso ajudou a captar insights qualitativos que não foram abordados nas questões anteriores, como sugestões de melhorias ou observações clínicas específicas que poderiam aumentar a utilidade do modelo.

4. RESULTADOS

Os resultados obtidos neste trabalho são baseados no desempenho do modelo somente no conjunto de testes do NIH *Chest X-ray Dataset*, com as 14 diferentes doenças relacionadas ao tórax. As figuras 18 e 19 mostram a distribuição do dataset em termos de número de casos por doença e de número de doenças em mesma imagem por quantidade de pacientes, respectivamente. Como perceptível na figura 18, doenças como *Infiltration*, *Effusion* e *Atelectasis* são as mais prevalentes, enquanto doenças como *Hernia*, Pneumonia e *Fibrosis* são consideravelmente mais raras. Já concernente à figura 19, salvo imagens com diagnóstico nulo, a maioria dos pacientes tem apenas uma doença, mas ainda havendo uma fração significativa com duas ou mais patologias concomitantes.

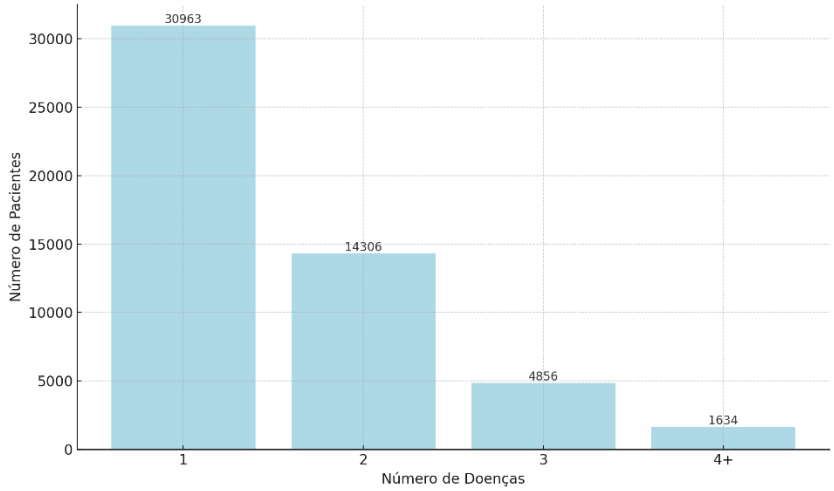
Esses dados refletem a heterogeneidade do *dataset* e são cruciais para compreender os resultados obtidos, visto que a presença de múltiplas patologias pode influenciar diretamente o desempenho do modelo e a interpretação das previsões.

Figura 18 – Número de casos por doença



Fonte: O autor.

Figura 19 – Número de doenças em mesma imagem por número de pacientes



Fonte: O autor.

4.1. Avaliação e Comparação por métrica AUC-ROC

Ao avaliar o desempenho do modelo utilizando a métrica AUC-ROC para cada doença, observamos variações consideráveis entre as patologias. Essas variações são esperadas devido à complexidade intrínseca de cada condição e à heterogeneidade das amostras disponíveis no *dataset*. A seguir, apresentamos os valores de AUC-ROC obtidos, comparando-os com estudos anteriores que utilizaram o mesmo NIH *Chest X-ray Dataset*. A ordem das doenças é a mesma introduzida na seção 2.3.

Tabela 1 – Síntese dos resultados em AUC-ROC por doença

Método	Atel	Card	Effu	Infi	Mass	Nodu	Pne1	Pne2	Cons	Edem	Emph	Fibr	P.T.	Hern	Média
Wang et al.	0.700	0.810	0.759	0.661	0.693	0.669	0.658	0.799	0.703	0.805	0.833	0.786	0.684	0.872	0.745
Guan et al. (ResNet)	0.779	0.879	0.824	0.694	0.831	0.766	0.726	0.858	0.758	0.850	0.909	0.832	0.778	0.906	0.814
Guan et al. (DenseNet)	0.781	0.883	0.831	0.697	0.830	0.764	0.725	0.866	0.758	0.853	0.911	0.826	0.780	0.918	0.816
Ma et al.	0.777	0.894	0.829	0.696	0.838	0.771	0.722	0.862	0.750	0.846	0.908	0.827	0.779	0.934	0.817
Hermoza et al.	0.775	0.881	0.831	0.695	0.826	0.789	0.741	0.879	0.747	0.846	0.936	0.833	0.793	0.917	0.821
Kim et al. (ResNet)	0.782	0.881	0.836	0.715	0.834	0.799	0.730	0.874	0.747	0.834	0.936	0.815	0.798	0.896	0.820
Kim et al. (DenseNet)	0.780	0.887	0.835	0.710	0.831	0.804	0.734	0.871	0.747	0.840	0.941	0.815	0.799	0.909	0.822
Proposto	0.775	0.886	0.833	0.702	0.824	0.794	0.727	0.869	0.744	0.844	0.929	0.831	0.784	0.929	0.819

Fonte: O autor.

Comparando os resultados obtidos com outros estudos que utilizaram o mesmo *dataset*, observamos que o desempenho do modelo proposto está alinhado ou acima da média dos modelos comparados, reforçando a qualidade e a eficácia da abordagem

adotada. Por exemplo, Kim *et al*. (2017) reportaram valores de AUC-ROC similares para condições como Cardiomegalia (Card) e Pneumotórax (Pne2) no uso do modelo DenseNet. Essas comparações indicam que o modelo DenseNet-50 utilizado neste trabalho é competitivo e eficaz na detecção das principais patologias nas radiografias de tórax.

4.2. Variabilidade e Performance

A análise dos valores de AUC-ROC demonstra que o modelo proposto atinge resultados competitivos em relação aos métodos existentes na literatura. As patologias com características visuais mais bem definidas apresentam os melhores desempenhos devido à clareza das suas manifestações nas radiografias. Em contrapartida, doenças com características visuais mais difusas e de natureza complexa, como Infiltração e Pneumonia, apresentam resultados mais modestos, porém ainda dentro da faixa de valores reportados por outros estudos.

A variabilidade nos valores de AUC-ROC entre as diferentes patologias reflete tal natureza visual de cada condição nas radiografias de tórax. A seguir, discute-se algumas das patologias com desempenhos diferenciados, oferecendo explicações objetivas baseadas nas características radiográficas de cada doença.

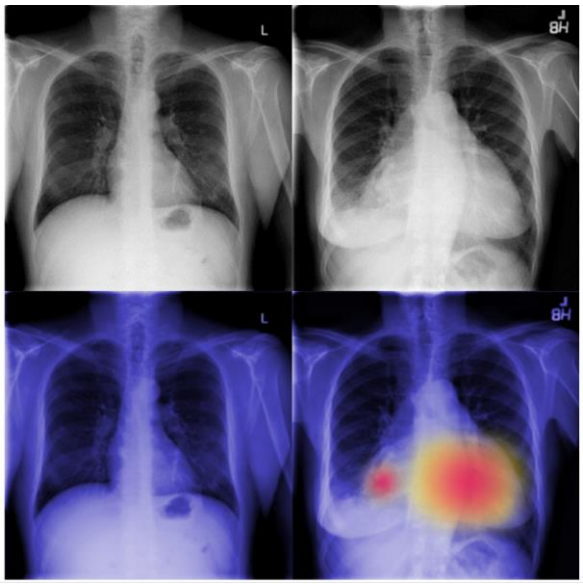
4.2.1. Hérnia (AUC-ROC: 0.929)

A Hernia apresenta um dos melhores desempenhos no modelo, com uma AUC-ROC de 0.929. Apesar de ter um número relativamente baixo de amostras no *dataset* (227 casos), sua apresentação anatômica distinta nas radiografias facilita a detecção. As hérnias geralmente manifestam-se como protrusões claras em regiões anatômicas específicas, tornando-as facilmente identificáveis tanto por profissionais de saúde quanto por modelos de *Deep Learning*.

4.2.2. Cardiomegalia (AUC-ROC: 0.886)

A Cardiomegalia é uma condição com desempenho robusto, refletindo sua manifestação visual proeminente nas radiografias. A cardiomegalia, ou aumento do tamanho do coração, é facilmente detectável devido ao aumento do contorno cardíaco e à sobreposição com outras estruturas torácicas, como os pulmões. Essa clareza visual facilita a identificação tanto por radiologistas quanto por algoritmos.

Figura 20 – Comparação de raios-x sem e com Cardiomegalia

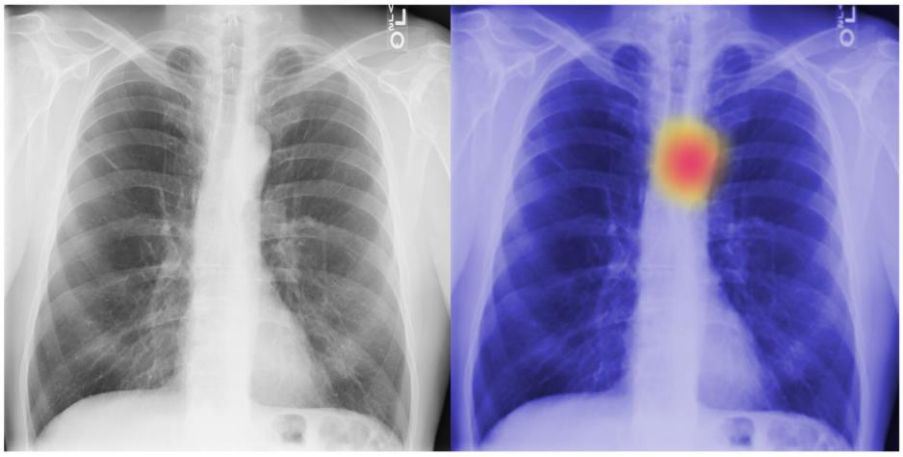


Fonte: O autor.

4.2.3. Infiltração (AUC-ROC: 0.702)

A Infiltração obteve um dos menores valores de AUC-ROC. Essa patologia apresenta características visuais mais difusas, como opacidades em vidro fosco ou consolidações leves que podem ser sutis e facilmente confundidas com artefatos ou outras condições benignas. A variabilidade na apresentação das infiltrações nas radiografias, associada à sobreposição com outras estruturas anatômicas, contribui para a dificuldade de detecção precisa pelo modelo.

Figura 21 – Não-percepção de Infiltração pelo Grad-CAM++



Fonte: O autor.

4.3. Doenças concomitantes e *Hamming Loss*

A *Hamming Loss* foi a métrica escolhida para avaliar o desempenho do modelo em um cenário de múltiplas doenças concomitantes, o que é uma característica relevante para o *dataset* utilizado, onde um único paciente pode apresentar mais de uma patologia ao mesmo tempo. Essa métrica mede a fração de rótulos incorretamente previstos, tanto os falsos positivos quanto os falsos negativos, sendo uma métrica comum para problemas de classificação *multilabel*.

A tabela com os resultados obtidos para a *Hamming Loss* está ilustrada abaixo, dividindo o conjunto de pacientes com base no número de doenças identificadas:

Tabela 2 – Métrica de *Hamming Loss* para uma ou mais doenças concomitantes

Métrica	1 Doença	2 Doenças	3 Doenças	4+ Doenças
Hamming Loss	0.175	0.257	0.308	0.360
1 - Hamming Loss	0.825	0.743	0.692	0.640

Fonte: O autor.

Os resultados indicam uma tendência clara: conforme o número de doenças concomitantes aumenta, o valor da *Hamming Loss* também aumenta. Esse comportamento é esperado, dado que a complexidade do diagnóstico cresce proporcionalmente à quantidade de doenças presentes em um único paciente.

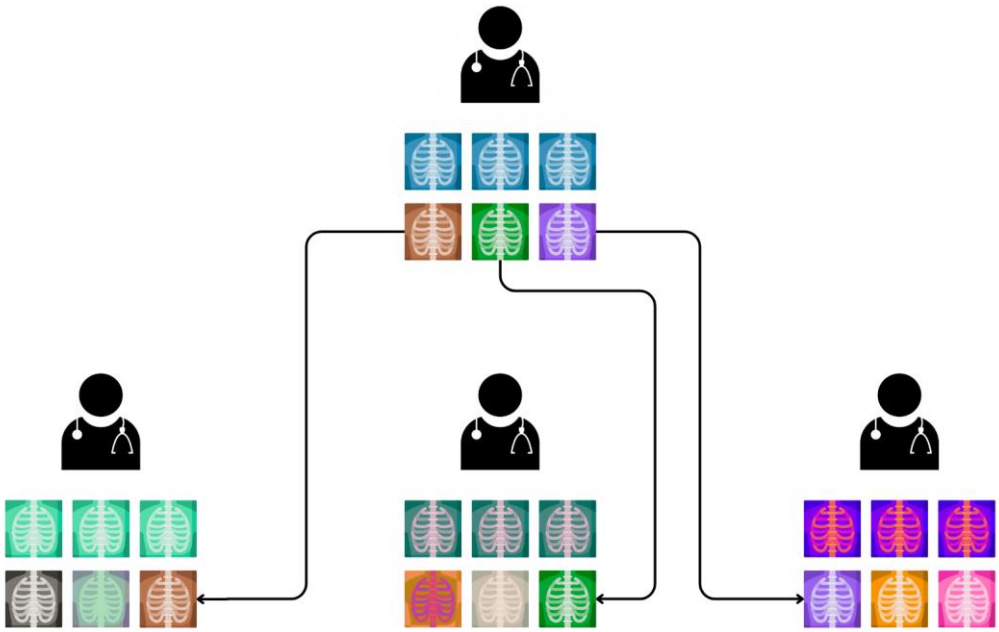
Os gráficos apresentados anteriormente mostram a distribuição do número de doenças por paciente e a frequência de doenças no *dataset*. A maior parte dos pacientes possui apenas uma ou duas doenças, o que explica o desempenho relativamente bom para esses casos em termos de *Hamming Loss*.

Pacientes com três ou mais doenças são menos frequentes no *dataset*, mas apresentam uma variabilidade clínica maior, com condições que podem se sobrepor em suas manifestações radiográficas. Isso contribui para a dificuldade de previsão do modelo, como evidenciado pela elevação da *Hamming Loss* para esses grupos.

4.4. Avaliação com Profissionais de Saúde

Ao todo, 12 profissionais da área (médicos radiologistas e técnicos radiologistas) concordaram em participar da validação do modelo. Como explicado anteriormente, cada profissional recebeu 6 imagens escolhidas aleatoriamente dentre as que continham ao menos uma doença, onde 3 dessas eram únicas e as outras 3 repetiam-se mais duas vezes, formando, então, 72 análises em 48 exemplos.

Figura 22 – Representação simplificada do esquema de avaliação



Fonte: O autor.

Devido ao número moderado de exemplos avaliados, a análise dos resultados foi realizada de forma global em vez de segmentada por patologia. Das 48 imagens, o modelo acertou 39. Abaixo está uma síntese dos resultados, onde o número entre parênteses representa o número de imagens:

Pergunta 1: Diagnóstico Visual (Imagem Original)

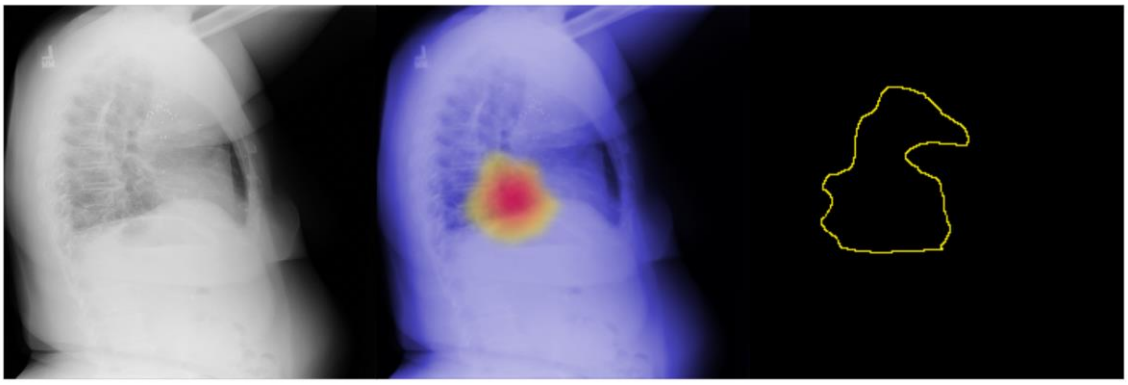
Resultado: Os profissionais acertaram 76.4% (55) das imagens originais, sendo os outros 23,6% (17) considerados como “inconclusivos”. Nenhum profissional deu um diagnóstico errado.

Após esses diagnósticos, foram reveladas as doenças originais de cada imagem.

Pergunta 2: Relevância do Mapa de Calor (Grad-CAM++)

Resultado: Os profissionais consideraram que 69.4% (50) das imagens geradas pelo Grad-CAM++ como clinicamente alinhadas às áreas relevantes do diagnóstico. Interessantemente, 9.7% (7) das imagens consideradas úteis foram consideradas como inconclusivas pelos profissionais, demonstrando que o modelo conseguiu performar acima dos diagnósticos profissionais em alguns casos.

Figura 23 – Raio-X inconclusivo pelos especialistas, mas positivo pelo modelo



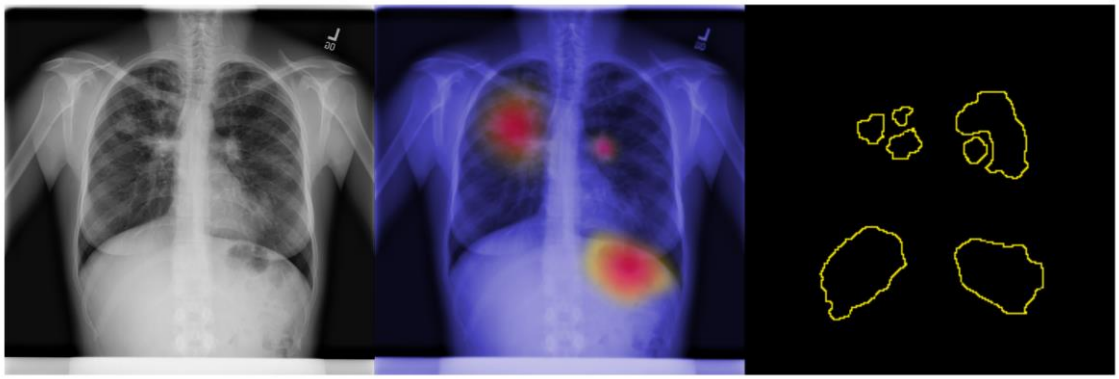
Fonte: O autor.

Pergunta 3: Precisão das Áreas Destacadas (LIME)

Resultado: Os profissionais consideraram que 62.5% (45) das imagens geradas pelo LIME demonstravam corretamente as áreas que os profissionais visualizaram para concluírem o diagnóstico. Outros 11.1% (8) foram considerados como úteis, mas que demarcaram certas incongruências ou áreas atípicas do diagnóstico.

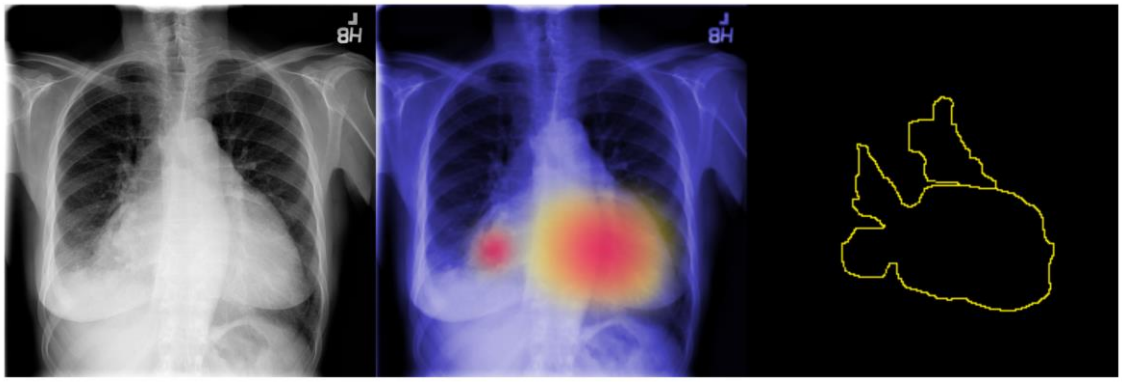
As figuras 24 e 25 representam diagnósticos presentes para 3 profissionais em que o LIME foi considerado como perfeito pelos 3 avaliadores. A figura 24 apresenta Nódulos e Espessamento Pleural, e a figura 25 apresenta Cardiomegalia e Infiltração.

Figura 24 – Raio-X de paciente com Nódulos e Espessamento Pleural



Fonte: O autor.

Figura 25 – Raio-X de paciente com Cardiomegalia e Infiltração



Fonte: O autor.

Pergunta 4: Grau de Confiabilidade (1 a 5)

Resultado: A média da confiabilidade foi de 4.3, indicando um nível elevado de confiança para com o modelo.

Essa avaliação numérica de confiabilidade sugere que os especialistas, em geral, se sentem seguros com as previsões e explicações fornecidas pelo modelo, e que, mesmo não obtendo resultados perfeitos em certos diagnósticos, o modelo mostrou-se útil em casos mais neblinados.

Pergunta 5: Comentários Adicionais

Em síntese, os profissionais ficaram impressionados pela assertividade e objetividade do modelo. Porém, em casos de múltiplas patologias, foi apontado que o modelo dificilmente separou as influências visuais de cada patologia, o que embora não comprometa tanto o diagnóstico, acaba mesmo assim sendo um ponto falho quanto à explicabilidade.

Ademais, a maioria dos profissionais destacou que as explicações visuais fornecidas pelo modelo têm potencial clínico, especialmente como uma ferramenta de suporte ao diagnóstico. Também houve comentários sugerindo que, em casos onde as previsões eram incertas, o modelo poderia beneficiar de uma maior integração de dados clínicos junto às imagens radiológicas para melhorar a acurácia.

5. CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um modelo de *Deep Learning* baseado na arquitetura DenseNet para o diagnóstico automático de múltiplas patologias em radiografias de tórax, com o suporte das técnicas de explicabilidade Grad-CAM++ e LIME. Ao longo do estudo, foi realizada uma série de experimentos que demonstraram a eficácia do modelo, especialmente em patologias mais evidentes, como Cardiomegalia, Enfisema, e Hérnia, que apresentaram valores de AUC-ROC elevados, confirmando a capacidade do modelo em identificar corretamente essas doenças.

A inclusão de explicações visuais por meio do Grad-CAM++ foi um diferencial importante, uma vez que possibilitou a geração de mapas de calor que realçavam as regiões das imagens que mais contribuíram para o diagnóstico. Esse tipo de abordagem, além de auxiliar no entendimento das previsões, proporcionou maior transparência ao modelo, um aspecto crucial para a adoção de redes neurais na prática clínica. O uso dessas técnicas de explicabilidade teve impacto positivo na confiança dos profissionais de saúde, conforme evidenciado pelo processo de avaliação realizado.

Outro ponto importante foi o uso do *Hamming Loss* para medir a precisão das previsões em casos de doenças concomitantes, um cenário comum na análise de radiografias de tórax. Observamos que a taxa de erro do modelo aumentava à medida que o número de patologias presentes no paciente aumentava, evidenciando o desafio inerente a diagnósticos complexos.

Portanto, as principais contribuições deste trabalho foram:

- Desenvolvimento de um modelo eficiente para o diagnóstico automático de múltiplas patologias;
- Utilização de técnicas de explicabilidade visual, como o Grad-CAM++, para aumentar a transparência das previsões e fomentar a confiança no modelo;
- Avaliação do modelo com profissionais de saúde, que forneceram feedback valioso sobre a aplicabilidade clínica do sistema.

Essas descobertas ressaltam o potencial impacto clínico das redes neurais na medicina, especialmente como ferramentas de suporte ao diagnóstico, oferecendo maior precisão e velocidade na análise de grandes volumes de dados de imagem, como

radiografias. O uso de técnicas de explicabilidade, como demonstrado, contribui para a aceitação dessas tecnologias em ambientes clínicos, facilitando a integração dessas ferramentas no cotidiano dos profissionais de saúde.

5.1. Limitações

Apesar dos resultados promissores, este trabalho apresenta algumas limitações que devem ser consideradas. A primeira e mais significativa foi o baixo número de profissionais de saúde envolvidos na avaliação. O modelo foi avaliado por um número limitado de especialistas, o que pode restringir a generalização dos resultados obtidos sobre a aceitação e utilidade do modelo em diferentes contextos clínicos. Para garantir uma avaliação mais robusta, seria ideal que mais profissionais fossem incluídos em futuras avaliações.

Outra limitação foi a complexidade das doenças concomitantes, um fator que aumentou a taxa de erro do modelo, conforme indicado pelos resultados de *Hamming Loss*. O modelo demonstrou dificuldades crescentes à medida que o número de patologias por paciente aumentava, indicando que abordagens mais sofisticadas podem ser necessárias para melhorar o desempenho em cenários de maior complexidade.

Além disso, houve a limitação computacional enfrentada durante o processamento do grande volume de dados, o que exigiu a utilização de técnicas de *chunking* e o treinamento em uma GPU Tesla T4. Embora essas soluções tenham sido suficientes para realizar os experimentos, a necessidade de recursos computacionais mais avançados pode ser uma barreira para a implementação prática em outras instituições com recursos mais limitados.

5.2. Trabalhos Futuros

Para trabalhos futuros, propõe-se algumas direções que podem expandir e refinar os resultados apresentados neste TCC:

1. Aprimoramento da Explicabilidade:
    - Explorar outras técnicas de explicabilidade, além do Grad-CAM++ e do LIME, para entender melhor as decisões do modelo e proporcionar explicações mais precisas em casos de doenças concomitantes.

2. Treinamento com Dados Clínicos Adicionais:
    - Integrar dados clínicos (como histórico médico e sintomas) ao modelo, além das radiografias, permitindo que o modelo faça previsões mais informadas e contextualizadas, o que pode aumentar a precisão diagnóstica em casos mais complexos.
3. Exploração de Arquiteturas Diferentes:
    - A investigação de outras arquiteturas de rede, como *transformers* visuais (ViTs), pode trazer novas perspectivas para melhorar o desempenho, especialmente em casos de múltiplas patologias.
4. Implementação Prática em Ambiente Clínico:
    - Outra possibilidade seria desenvolver um sistema piloto para ser testado em ambientes clínicos reais, permitindo que o modelo seja usado como uma ferramenta de apoio no dia a dia dos profissionais e gerando dados reais de usabilidade e impacto.

5.3. Considerações Finais

Este trabalho demonstrou o potencial de redes neurais e técnicas de explicabilidade no diagnóstico automatizado de patologias em radiografias de tórax. Com os avanços futuros propostos, há um caminho promissor para que essas ferramentas possam ser efetivamente integradas à prática clínica, contribuindo para diagnósticos mais rápidos, precisos e confiáveis.

REFERÊNCIAS

CHATTOPADHYAY, A.; SARKAR, A.; HOWLADER, P.; BALASUBRAMANIAN, V. N. Grad-CAM++: improved visual explanations for deep convolutional networks. In: IEEE WINTER CONFERENCE ON APPLICATIONS OF COMPUTER VISION (WACV), 2018, Anaheim, CA. Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV). p. 839-847.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: a large-scale hierarchical image database. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2009, Miami, FL. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. p. 248–255. IEEE, 2009.

FAWCETT, T. An introduction to ROC analysis. *Pattern Recognition Letters*, v. 27, n. 8, p. 861–874, 2006.

FOLK, M.; HEBER, G.; KOZIOL, Q.; POURMAL, E.; ROBINSON, D. An overview of the HDF5 technology suite and its applications. In: EDBT/ICDT 2011 WORKSHOP ON ARRAY DATABASES, 2011, [Local]. Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases. p. 36-47.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge: MIT Press, 2016.

GOOGLE. Google Drive. Disponível em: https://www.google.com/drive/. Acesso em: [12 de agosto de 2024].

GUAN, Q.; HUANG, Y.; ZHONG, Z.; ZHENG, Z.; ZHENG, L.; YANG, Y. Diagnose like a radiologist: attention guided convolutional neural network for thorax disease classification. *arXiv preprint* arXiv:1801.09927, 2018.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, Las Vegas, NV. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. p. 770–778.

HUANG, G.; LIU, Z.; VAN DER MAATEN, L.; WEINBERGER, K. Q. Densely connected convolutional networks. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2017, Honolulu, HI. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. p. 4700-4708.

IRVIN, J.; RAJPURKAR, P.; KO, M.; YU, Y.; CIUREA-ILCUS, S.; CHUTE, C.; ... NG, A. Y. CheXpert: a large chest radiograph dataset with uncertainty labels and expert comparison. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, n. 01, p. 590-597, 2019.

JAPKOWICZ, N.; STEPHEN, S. The class imbalance problem: a systematic study. *Intelligent Data Analysis*, v. 6, n. 5, p. 429–449, 2002.

KIM, J.; RHEE, S.; LEE, Y. J.; PARK, S. H. Challenges in clinical implementation of AI models: explanation and reproducibility. *Journal of Korean Medical Science*, v. 35, n. 38, 2020.

KINGMA, D. P.; BA, J. Adam: a method for stochastic optimization. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR), 2015, [Local]. Proceedings of the 3rd International Conference on Learning Representations. 2015.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1995, Boston, MA. Proceedings of the 14th International Joint Conference on Artificial Intelligence. p. 1137–1145.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278-2324, 1998.

LUNDERVOLD, A. S.; LUNDERVOLD, A. An overview of deep learning in medical imaging focusing on MRI. *Zeitschrift für Medizinische Physik*, v. 29, n. 2, p. 102-127, 2019.

MAJKOWSKA, A.; MITTAL, S.; STEINER, D. F.; REICHER, J. J.; MCKINNEY, S. M.; DUGGAN, G. E.; ... KALPATHY-CRAMER, J. Chest radiograph interpretation with

deep learning models: assessment with radiologist-adjudicated reference standards and population-adjusted evaluation. *Radiology*, v. 294, n. 2, p. 421–431, 2020.

NG, A. Y. Feature selection, L1 vs. L2 regularization, and rotational invariance. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2004, Honolulu, HI. Proceedings of the 21st International Conference on Machine Learning. p. 78.

NIH CLINICAL CENTER. NIH Chest X-ray Dataset of 14 Common Thorax Disease Categories. Disponível em: https://nihcc.app.box.com/v/ChestXray-NIHCC. Acesso em: [14 de agosto de 2024].

NVIDIA. NVIDIA Tesla T4 GPU Architecture. Whitepaper, 2018.

PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint* arXiv:1712.04621, 2017.

PRECHELT, L. Early stopping—But when? In: NEURAL NETWORKS: TRICKS OF THE TRADE. Neural Networks: Tricks of the trade. Springer, p. 55–69, 1998.

RAJPURKAR, P.; IRVIN, J.; ZHU, K.; YANG, B.; MEHTA, H.; DUAN, T.; ... NG, A. Y. CheXNet: radiologist-level pneumonia detection on chest X-rays with deep learning. *arXiv preprint* arXiv:1711.05225, 2017.

SAMEK, W.; WIEGAND, T.; MÜLLER, K. R. Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models. *arXiv preprint* arXiv:1708.08296, 2017.

SCHAPIRE, R. E.; SINGER, Y. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, v. 39, n. 2-3, p. 135–168, 2000.

SELVARAJU, R. R.; COGSWELL, M.; DAS, A.; VEDANTAM, R.; PARIKH, D.; BATRA, D. Grad-CAM: visual explanations from deep networks via gradient-based localization. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2017, Venice, Italy. Proceedings of the IEEE International Conference on Computer Vision. p. 618-626.

SHIN, H.-C.; ROTH, H. R.; GAO, M.; LU, L.; XU, Z.; NOGUES, I.; ... SUMMERS, R. M. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, v. 35, n. 5, p. 1285–1298, 2016.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014.

TJOA, E.; GUAN, C. A survey on explainable artificial intelligence (XAI): toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

WANG, X.; PENG, Y.; LU, L.; LU, Z.; BAGHERI, M.; SUMMERS, R. M. ChestX-ray8: hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2017, Honolulu, HI. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. p. 2097-2106.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? In: NEURAL INFORMATION PROCESSING SYSTEMS (NeurIPS), 2014, Montreal, Canada. Proceedings of the 27th International Conference on Neural Information Processing Systems. p. 3320–3328.

ZHANG, Q.; WU, Y. N.; ZHU, S. C. Interpretable convolutional neural networks. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2018, Salt Lake City, UT. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). p. 8827-8836.

APÊNDICE – JUPYTER NOTEBOOK DO CÓDIGO DO TRABALHO

```
# Montagem do Google Drive para acesso aos dados
from google.colab import drive
drive.mount('/content/drive')

# Importação das bibliotecas necessárias
import os
import cv2
import numpy as np
import h5py
from multiprocessing.pool import ThreadPool
import pandas as pd

# Caminho para o arquivo CSV com os rótulos
csv_full_path =
'/content/drive/MyDrive/MeuProjeto/Data_Entry_2017.csv'

# Carregar o CSV completo com os rótulos
df_full = pd.read_csv(csv_full_path)

# Definição das 14 doenças presentes no dataset
diseases = [
    'Atelectasis', 'Cardiomegaly', 'Effusion', 'Infiltration',
'Mass',
    'Nodule', 'Pneumonia', 'Pneumothorax', 'Consolidation',
'Edema',
    'Emphysema', 'Fibrosis', 'Pleural_Thickening', 'Hernia'
]

# Função para converter a string de rótulos em um vetor binário
def get_binary_vector(labels):
    binary_vector = np.zeros(len(diseases), dtype=int)
    labels_list = labels.split('|')  # Divide caso haja
múltiplas doenças separadas por '|'
    for idx, disease in enumerate(diseases):
        if disease in labels_list:
            binary_vector[idx] = 1
    return binary_vector

# Construção do dicionário de rótulos
labels_dict = {}
for index, row in df_full.iterrows():
    image_name = row['Image Index']
    labels = row['Finding Labels']
    binary_vector = get_binary_vector(labels)
    labels_dict[image_name] = binary_vector
```

```
print(f"Número total de rótulos no labels_dict:
{len(labels_dict)}")

# Função para processar e normalizar uma única imagem
def process_single_image(img_path, img_size):
    """
    Processa uma única imagem: leitura, redimensionamento e
normalização.
    """
    img = cv2.imread(img_path)
    if img is not None:
        img = cv2.resize(img, img_size)
        img = img / 255.0  # Normaliza os valores dos pixels
    return img

# Função para salvar um chunk de imagens e rótulos em um arquivo
HDF5 no Google Drive
def save_images_and_labels_to_chunk(images, image_names, labels,
chunk_count, save_dir='/content/drive/MyDrive/chunks'):
    """
    Salva um chunk de imagens e rótulos em um arquivo HDF5 no
Google Drive.
    """
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    save_path = os.path.join(save_dir,
f'chunk_{chunk_count}.h5')

    # Converter os nomes das imagens para bytes (utf-8) para
serem salvos corretamente
    image_names_bytes = [name.encode('utf-8') for name in
image_names]

    with h5py.File(save_path, 'w') as f:
        f.create_dataset('images', data=images)
        f.create_dataset('image_names', data=image_names_bytes)
# Salvando os nomes das imagens como bytes
        f.create_dataset('labels', data=labels)

    print(f"Chunk {chunk_count} salvo com {len(images)}
imagens.")  # Exibe uma mensagem quando um chunk é salvo

# Função para carregar e processar imagens em chunks
def load_images_and_save_in_chunks(main_dir, labels_dict,
file_list, chunk_size=10000, img_size=(224, 224),
num_threads=4):
    """
    Carrega imagens em chunks e as salva no disco para evitar
problemas de memória, com processamento paralelo.
```

```
    """
    images = []
    image_names = []
    labels = []
    processed_images = 0
    chunk_count = 1

    # Usar ThreadPool para processar imagens em paralelo
    pool = ThreadPool(processes=num_threads)

    # Percorrer todas as pastas (images_001, images_002, etc.)
    for subdir in os.listdir(main_dir):
        subdir_path = os.path.join(main_dir, subdir, 'images')
# Caminho para a subpasta "images"
        if os.path.exists(subdir_path):  # Verifica se a
subpasta 'images' existe

            image_paths = []
            valid_image_names = []

            # Coletar as imagens válidas para processamento
            for image_name in os.listdir(subdir_path):
                if image_name in file_list and
image_name.lower().endswith(('.png', '.jpg', '.jpeg')):
                    img_path = os.path.join(subdir_path,
image_name)
                    image_paths.append(img_path)
                    valid_image_names.append(image_name)

            # Processar as imagens em paralelo usando ThreadPool
            if image_paths:
                results = pool.map(lambda p:
process_single_image(p, img_size), image_paths)

                # Adicionar as imagens processadas e seus
respectivos nomes e rótulos
                for i, img in enumerate(results):
                    if img is not None:
                        images.append(img)
                        image_names.append(valid_image_names[i])

                        # Obter o rótulo para a imagem usando
seu nome
                        if valid_image_names[i] in labels_dict:

labels.append(labels_dict[valid_image_names[i]])

                        processed_images += 1
```

```
                        # Quando atingirmos o tamanho do chunk,
salvamos e limpamos as listas
                        if processed_images % chunk_size == 0:

save_images_and_labels_to_chunk(np.array(images),
np.array(image_names), np.array(labels), chunk_count)
                            chunk_count += 1
                            images, image_names, labels = [],
[], []  # Limpar listas para o próximo chunk

                        # Atualizar o progresso a cada 100
imagens processadas
                        if processed_images % 100 == 0:
                            print(f"{processed_images} imagens
processadas.")

    # Salvar qualquer sobra de imagens no último chunk
    if images:
        save_images_and_labels_to_chunk(np.array(images),
np.array(image_names), np.array(labels), chunk_count)

    print(f"Processamento completo: {processed_images} imagens
processadas.")
    return processed_images

# Função para carregar a lista de arquivos a partir dos arquivos
.txt
def load_file_list(file_path):
    with open(file_path, 'r') as file:
        return [line.strip() for line in file]

# Caminhos para os arquivos txt no Google Drive
train_val_list_path =
'/content/drive/MyDrive/MeuProjeto/train_val_list_reduced.txt'
test_list_path =
'/content/drive/MyDrive/MeuProjeto/test_list_reduced.txt'

# Carregar as listas de treino/validação e teste
train_val_list = load_file_list(train_val_list_path)
test_list = load_file_list(test_list_path)

print(f"Número de imagens de treino/validação:
{len(train_val_list)}")
print(f"Número de imagens de teste: {len(test_list)}")

# Caminho da pasta principal com as subpastas de imagens no
Google Drive
image_dir = '/content/drive/MyDrive/MeuProjeto'
```

```
# Processar e salvar as imagens de treino/validação em chunks
total_processed_images =
load_images_and_save_in_chunks(image_dir, labels_dict,
train_val_list, chunk_size=10000)

print(f"Total de imagens processadas e salvas em chunks:
{total_processed_images}")

# Processar e salvar as imagens de teste em chunks
total_processed_test_images =
load_images_and_save_in_chunks(image_dir, labels_dict,
test_list, chunk_size=10000)

print(f"Total de imagens de teste processadas e salvas em
chunks: {total_processed_test_images}")
```

```
import os
import h5py
import numpy as np

# Caminho para a pasta onde os chunks estão salvos
chunk_dir = '/content/drive/MyDrive/chunks'

# Listar todos os chunks existentes
chunks = sorted([os.path.join(chunk_dir, f) for f in
os.listdir(chunk_dir) if f.endswith('.h5')])

# Função para atualizar os rótulos em um chunk existente
def update_labels_in_chunk(chunk_path, labels_dict):
    with h5py.File(chunk_path, 'r+') as f:  # Usar 'r+' para
leitura e escrita
        # Carregar os nomes das imagens
        image_names_bytes = f['image_names'][:]
        # Converter os nomes das imagens de bytes para strings
        image_names = [name.decode('utf-8') for name in
image_names_bytes]

        # Se o dataset 'labels' existir, deletá-lo
        if 'labels' in f:
            print(f"Deletando o dataset 'labels' existente no
chunk: {chunk_path}")
            del f['labels']

        # Criar a lista de rótulos para as imagens do chunk
        labels = []
        missing_labels = 0
        for image_name in image_names:
```

```
            if image_name in labels_dict:
                labels.append(labels_dict[image_name])
            else:
                print(f"Aviso: Rótulo não encontrado para
{image_name}")
                labels.append(np.zeros(len(diseases),
dtype=int))  # Adiciona vetor zero se não encontrar
                missing_labels += 1

        if missing_labels > 0:
            print(f"{missing_labels} imagens sem rótulos
encontradas no chunk {chunk_path}")

        # Converter para numpy array
        labels = np.array(labels)

        # Adicionar os rótulos ao chunk
        f.create_dataset('labels', data=labels)
        print(f"Rótulos atualizados no chunk {chunk_path} com
sucesso!")

# Atualizar os rótulos em todos os chunks
for chunk in chunks:
    update_labels_in_chunk(chunk, labels_dict)

# Função para verificar um chunk
def verify_chunk(chunk_path):
    with h5py.File(chunk_path, 'r') as f:
        images = f['images']
        labels = f['labels']
        image_names = f['image_names']
        print(f"Chunk: {chunk_path}")
        print(f"Número de imagens: {len(images)}")
        print(f"Número de rótulos: {len(labels)}")
        print(f"Exemplo de rótulo: {labels[50]}")
        print("-" * 50)

# Verificar todos os chunks
for chunk in chunks:
    verify_chunk(chunk)

import numpy as np
import h5py
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import
```

```
ImageDataGenerator

# Função para carregar um chunk de imagens e rótulos
def load_chunk(chunk_path):
    with h5py.File(chunk_path, 'r') as f:
        images = np.array(f['images'])
        labels = np.array(f['labels'])
    return images, labels
```

```
from tensorflow.keras.applications import DenseNet50
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

def create_model(num_classes):
    # Carregar a DenseNet-50 pré-treinada no ImageNet, sem as
camadas superiores
    base_model = DenseNet50(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))

    # Congelar as camadas da DenseNet-50 para usar Transfer
Learning
    for layer in base_model.layers:
        layer.trainable = False

    # Adicionar camadas adicionais para adaptação ao nosso
problema
    x = base_model.output
    x = GlobalAveragePooling2D()(x)  # Pooling global
    x = Dense(1024, activation='relu')(x)  # Camada densa
adicional
    x = Dropout(0.5)(x)  # Camada de Dropout com taxa de 50%

    # Camada de saída para classificação multilabel (14 classes)
com ativação sigmoide
    predictions = Dense(num_classes, activation='sigmoid')(x)

    # Criar o modelo final
    model = Model(inputs=base_model.input, outputs=predictions)

    # Compilar o modelo com otimizador Adam e loss binary
crossentropy para multilabel
    model.compile(optimizer=Adam(learning_rate=1e-4),
loss='binary_crossentropy', metrics=['auc'])

    return model
```

```
# Criação do modelo para 14 classes
model = create_model(num_classes=14)
model.summary()

from sklearn.model_selection import KFold
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint
import numpy as np
import os

# Número de folds
k = 5
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Caminho para a pasta onde os chunks estão salvos
chunk_dir = '/content/drive/MyDrive/chunks'
chunks = sorted([os.path.join(chunk_dir, f) for f in
os.listdir(chunk_dir) if f.endswith('.h5')])

# Lista completa de imagens e rótulos para todos os chunks
all_images = []
all_labels = []

for chunk in chunks:
    images, labels = load_chunk(chunk)
    all_images.append(images)
    all_labels.append(labels)

all_images = np.concatenate(all_images, axis=0)
all_labels = np.concatenate(all_labels, axis=0)

# Inicializar a validação cruzada
fold_no = 1
all_histories = []

for train_index, val_index in kf.split(all_images):
    print(f"\nIniciando o Fold {fold_no}/{k}")

    # Dividir os dados em treino e validação
    X_train, X_val = all_images[train_index],
all_images[val_index]
    y_train, y_val = all_labels[train_index],
all_labels[val_index]

    # Criar o modelo
    model = create_model(num_classes=14)

    # Definir callbacks
```

```
    early_stop = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
    checkpoint =
ModelCheckpoint(f'/content/drive/MyDrive/MeuProjeto/model_fold_{
fold_no}.h5',
                                    monitor='val_auc',
                                    mode='max',
                                    save_best_only=True,
                                    verbose=1)

    # Configurar o gerador de dados com Data Augmentation para o
treino
    train_datagen = ImageDataGenerator(
        rotation_range=15,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True
    )

    val_datagen = ImageDataGenerator()  # Apenas normalização
para validação

    # Geradores
    train_generator = train_datagen.flow(X_train, y_train,
batch_size=32)
    val_generator = val_datagen.flow(X_val, y_val,
batch_size=32)

    # Treinar o modelo
    history = model.fit(
        train_generator,
        epochs=30,
        validation_data=val_generator,
        callbacks=[early_stop, checkpoint],
        verbose=1
    )

    # Armazenar o histórico
    all_histories.append(history.history)

    # Incrementar o número do fold
    fold_no += 1

    # Liberar recursos
    del model, train_generator, val_generator
    gc.collect()

print("\nValidação Cruzada Completa.")
```

```
from sklearn.metrics import classification_report, hamming_loss,
roc_auc_score

# Avaliar o modelo em um conjunto de teste separado (já
preparado anteriormente)

# Fazer previsões no conjunto de teste
y_pred = model.predict(test_images)
y_pred_bin = (y_pred > 0.5).astype(int)  # Converter as
previsões para 0 e 1 (binarização)

# Avaliar o modelo usando várias métricas
print("Classification Report:")
print(classification_report(test_labels, y_pred_bin,
target_names=diseases))

print(f"Hamming Loss: {hamming_loss(test_labels, y_pred_bin)}")

# AUC-ROC Score para cada classe
roc_auc = roc_auc_score(test_labels, y_pred, average=None)
for idx, disease in enumerate(diseases):
    print(f"AUC-ROC para {disease}: {roc_auc[idx]}")

import matplotlib.pyplot as plt

def plot_kfold_history(all_histories, k):
    # Consolidar os históricos
    avg_loss = np.mean([history['loss'] for history in
all_histories], axis=0)
    avg_val_loss = np.mean([history['val_loss'] for history in
all_histories], axis=0)
    avg_auc = np.mean([history['auc'] for history in
all_histories], axis=0)
    avg_val_auc = np.mean([history['val_auc'] for history in
all_histories], axis=0)

    epochs = range(1, len(avg_loss) + 1)

    # Plotar a perda
    plt.figure(figsize=(14, 6))

    plt.subplot(1, 2, 1)
    plt.plot(epochs, avg_loss, 'b-', label='Perda de Treino')
    plt.plot(epochs, avg_val_loss, 'r-', label='Perda de
Validação')
    plt.title('Perda durante o Treinamento')
    plt.xlabel('Épocas')
    plt.ylabel('Perda')
```

```
    plt.legend()

    # Plotar a AUC-ROC
    plt.subplot(1, 2, 2)
    plt.plot(epochs, avg_auc, 'b-', label='AUC-ROC de Treino')
    plt.plot(epochs, avg_val_auc, 'r-', label='AUC-ROC de
Validação')
    plt.title('AUC-ROC durante o Treinamento')
    plt.xlabel('Épocas')
    plt.ylabel('AUC-ROC')
    plt.legend()

    plt.show()

# Chamar a função para plotar os gráficos
plot_kfold_history(all_histories, k=5)
```

```
from tensorflow.keras.models import load_model

# Salvar o modelo treinado
model.save('/content/drive/MyDrive/MeuProjeto/modelo_treinado.h5
')
print("Modelo salvo com sucesso.")

# Carregar o modelo salvo
model_path =
'/content/drive/MyDrive/MeuProjeto/modelo_treinado.h5'
model = load_model(model_path)
print("Modelo carregado com sucesso.")
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import cv2

def grad_cam_plus_plus(model, image, class_idx,
layer_name='conv5_block3_out'):
    """
    Gera um mapa de calor Grad-CAM++ para uma imagem e uma
classe específica.

    Args:
        model: Modelo treinado.
        image: Imagem de entrada pré-processada (numpy array de
forma (224, 224, 3)).
        class_idx: Índice da classe para a qual gerar o mapa de
calor.
        layer_name: Nome da camada convolucional de interesse.
```

```
    Returns:
        heatmap: Mapa de calor Grad-CAM++ normalizado.
    """
    # Preparar a imagem adicionando a dimensão do batch
    inputs = tf.expand_dims(image, axis=0)  # Forma: (1, 224,
224, 3)
    inputs = tf.cast(inputs, tf.float32)

    # Criar um modelo que retorna as ativações da camada de
interesse e as previsões
    grad_model = tf.keras.models.Model(
        [model.inputs], [model.get_layer(layer_name).output,
model.output]
    )

    with tf.GradientTape(persistent=True) as tape1:
        with tf.GradientTape(persistent=True) as tape2:
            with tf.GradientTape() as tape3:
                # Forward pass
                conv_outputs, predictions = grad_model(inputs)
                loss = predictions[:, class_idx]
            # Primeira derivada
            grads = tape3.gradient(loss, conv_outputs)
        # Segunda derivada
        first_derivative = grads
        second_derivative = tape2.gradient(first_derivative,
conv_outputs)
    # Terceira derivada
    third_derivative = tape1.gradient(second_derivative,
conv_outputs)
    del tape1
    del tape2
    del tape3

    # Remover a dimensão do batch
    conv_outputs = conv_outputs[0]  # Forma: (7, 7, 1024)
    grads = grads[0]  # Forma: (7, 7, 1024)
    first_derivative = first_derivative[0]  # Forma: (7, 7,
1024)
    second_derivative = second_derivative[0]  # Forma: (7, 7,
1024)
    third_derivative = third_derivative[0]  # Forma: (7, 7,
1024)

    # Cálculo dos pesos alpha
    alpha_num = second_derivative
    alpha_denom = 2 * second_derivative * conv_outputs.numpy() +
third_derivative * conv_outputs.numpy() ** 2
    alpha_denom = np.where(alpha_denom != 0.0, alpha_denom,
```

```
np.ones_like(alpha_denom))
    alphas = alpha_num / alpha_denom

    # Cálculo dos pesos de importância
    weights = np.maximum(grads, 0.0)
    deep_linearization_weights = np.sum(alphas * weights,
axis=(0, 1))

    # Cálculo do Grad-CAM++
    cam = np.sum(deep_linearization_weights *
conv_outputs.numpy(), axis=-1)

    # Redimensionar o mapa de calor para as dimensões da imagem
original
    cam = cv2.resize(cam, (image.shape[1], image.shape[0]))
    cam = np.maximum(cam, 0)
    heatmap = cam / np.max(cam) if np.max(cam) != 0 else cam

    return heatmap

def display_gradcam_plus_plus(image, heatmap, alpha=0.4):
    """
    Superpõe o mapa de calor Grad-CAM++ na imagem original e
exibe.

    Args:
        image: Imagem original pré-processada (numpy array de
forma (224, 224, 3)).
        heatmap: Mapa de calor Grad-CAM++ normalizado.
        alpha: Transparência do mapa de calor sobre a imagem
original.
    """
    # Converter a imagem para RGB se estiver em BGR
    if image.shape[-1] == 3:
        image_rgb = cv2.cvtColor((image * 255).astype(np.uint8),
cv2.COLOR_BGR2RGB)
    else:
        image_rgb = image

    # Aplicar a coloração no heatmap
    heatmap_colored = cv2.applyColorMap(np.uint8(255 * heatmap),
cv2.COLORMAP_JET)

    # Superpor o heatmap à imagem original
    superimposed_img = heatmap_colored * alpha + image_rgb
    superimposed_img = np.clip(superimposed_img, 0,
255).astype(np.uint8)
```

```
    # Exibir a imagem
    plt.figure(figsize=(8, 8))
    plt.imshow(superimposed_img)
    plt.axis('off')
    plt.show()

# Exemplo de aplicação do Grad-CAM++ em uma imagem de teste
class_idx = 0  # Índice da classe de interesse (exemplo:
'Atelectasis')
image = test_images[1]  # Certifique-se de que a imagem esteja
pré-processada

# Gerar o mapa de calor Grad-CAM++
heatmap = grad_cam_plus_plus(model, image, class_idx)

# Exibir o mapa de calor sobre a imagem original
display_gradcam_plus_plus(image, heatmap)
```

```
# Instalar a biblioteca LIME (executar apenas uma vez)
!pip install lime

import lime
from lime import lime_image
from skimage.segmentation import mark_boundaries
import matplotlib.pyplot as plt
import numpy as np

# Função de previsão adaptada para o LIME
def predict_fn(images):
    """
    Função de previsão para o LIME.

    Args:
        images: Array de imagens no formato [batch_size, height,
width, channels].

    Returns:
        Array de probabilidades para cada classe.
    """
    return model.predict(images)

# Criar o explicador LIME
explainer = lime_image.LimeImageExplainer()

# Selecionar uma imagem de teste para explicação
image_idx = 1  # Índice da imagem no conjunto de teste
image = test_images[image_idx]  # Imagem pré-processada
true_labels = test_labels[image_idx]  # Rótulos verdadeiros
```

```
# Obter as previsões do modelo
predictions = model.predict(np.expand_dims(image, axis=0))
predicted_class = np.argmax(predictions[0])  # Classe com maior
probabilidade

# Explicar a previsão para a classe prevista
explanation = explainer.explain_instance(
    image,                # A imagem a ser explicada
    predict_fn,           # A função de previsão
    top_labels=5,         # Número de classes para explicar
    hide_color=0,         # Cor de fundo para superpixels
escondidos
    num_samples=1000      # Número de amostras para a explicação
)

# Obter a explicação para a classe de interesse
temp, mask = explanation.get_image_and_mask(
    predicted_class,          # Classe a ser explicada
    positive_only=True,       # Mostrar apenas superpixels
positivos
    num_features=10,          # Número de superpixels a destacar
    hide_rest=False            # Não esconder o restante da
imagem
)

# Exibir a explicação LIME
plt.figure(figsize=(8, 8))
plt.imshow(mark_boundaries(temp / 255.0, mask))
plt.title(f"Explicação LIME para a classe:
{diseases[predicted_class]}")
plt.axis('off')
plt.show()
```

```
import pandas as pd
from collections import Counter

# Carregar o CSV
csv_path =
'/content/drive/MyDrive/MeuProjeto/Data_Entry_2017.csv'
df = pd.read_csv(csv_path)

# Visualizar as primeiras linhas do dataset para verificar a
estrutura
print(df.head())

# A coluna 'Finding Labels' contém as doenças, separadas por '|'
df['Finding Labels'] = df['Finding Labels'].fillna('')  #
```

```
Preencher NaN com string vazia

# Contar a quantidade de ocorrências de cada doença
disease_counts = Counter()
for labels in df['Finding Labels']:
    diseases = labels.split('|')  # Dividir as doenças por '|'
    for disease in diseases:
        if disease:  # Evitar contar strings vazias
            disease_counts[disease] += 1

# Exibir a quantidade de cada doença
print("Contagem de cada doença:")
for disease, count in disease_counts.items():
    print(f"{disease}: {count}")

# Contar quantos pacientes têm 0, 1, 2, 3, ... doenças
disease_distribution = Counter()
for labels in df['Finding Labels']:
    num_diseases = len([disease for disease in labels.split('|')
if disease])  # Contar doenças não vazias
    disease_distribution[num_diseases] += 1

# Exibir a distribuição de pacientes por número de doenças
print("\nDistribuição de pacientes por número de doenças:")
for num_diseases, count in sorted(disease_distribution.items()):
    print(f"{num_diseases} doenças: {count} pacientes")
```