UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MESTRADO ACADÊMICO EM ENGENHARIA ELÉTRICA

ARTUR GOMES BARRETO

UM NOVO MODELO GENERATIVO PARA DESCRIÇÕES TEXTUAIS DE IMAGENS MÉDICAS UTILIZANDO TRANSFORMADORES REALÇADOS COM REDES NEURAIS CONVOLUCIONAIS

FORTALEZA

2023

ARTUR GOMES BARRETO

UM NOVO MODELO GENERATIVO PARA DESCRIÇÕES TEXTUAIS DE IMAGENS MÉDICAS UTILIZANDO TRANSFORMADORES REALÇADOS COM REDES NEURAIS CONVOLUCIONAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica. Área de Concentração: Sistemas de Energia Elétrica.

Orientador: Prof. Dr. Victor Hugo Costa de Albuquerque.

FORTALEZA

2023

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)

B26n Barreto, Artur Gomes.
Um Novo Modelo Generativo para Descrições Textuais de Imagens Médicas Utilizando Transformadores Realçados com Redes Neurais Convolucionais / Artur Gomes Barreto. – 2023.
130 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2023.
Orientação: Prof. Dr. Victor Hugo Costa de Albuquerque.

1. Processamento Digital de Imagem. 2. Processamento de Linguagem Natural. 3. Transferência de Conhecimento. 4. Engenharia Biomédica. 5. ROCO. I. Título.

CDD 621.3

ARTUR GOMES BARRETO

UM NOVO MODELO GENERATIVO PARA DESCRIÇÕES TEXTUAIS DE IMAGENS MÉDICAS UTILIZANDO TRANSFORMADORES REALÇADOS COM REDES NEURAIS CONVOLUCIONAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica. Área de Concentração: Sistemas de Energia Elétrica.

Aprovada em: 01/09/2023.

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
VICTOR HUGO COSTA DE ALBUQUERQUE
Data: 06/09/2023 13:16:57-0300
Verifique em https://validar.iti.gov.br

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

FRANCISCO NAUBER BERNARDO GOIS:74397176353
Assinado de forma digital por FRANCISCO NAUBER BERNARDO GOIS:74397176353
Dados: 2023.09.06 08:37:57 -03'00'

Prof. Dr. Francisco Nauber Bernardo Gois
Controladoria Geral do Estado (CGE)

gov.br Documento assinado digitalmente
JULIANA MARTINS DE OLIVEIRA
Data: 06/09/2023 20:28:09-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Juliana Martins de Oliveira
Universidade de Fortaleza (Unifor)

Aos meus pais, Adonias (*in memoriam*) e Elda.

AGRADECIMENTOS

A minha esposa Grazielen Alencar Lima pelo companheirismo, seu amor e por seu apoio e paciência.

Aos meus pais, Adonias Freires Barreto e Elda Gomes Barreto, por terem me dado uma boa educação, não apenas a que adquiri nas boas escolas e nas universidades que frequentei, mas aquela que se aprende em casa, norteada por princípios universais da ética.

Aos meus irmãos, Sandra Barreto Dantas e Marcelo Gomes Barreto, por sempre estarem ao meu lado, me apoiando em meus desafios.

Ao Prof. Dr. Victor Hugo Costa de Albuquerque por me orientar no extraordinário e desafiante mundo da pós-graduação.

Aos professores participantes da banca examinadora Prof. Dr. Francisco Nauber Bernardo Gois e Profa. Dra. Juliana Martins de Oliveira pelo tempo concedido para a avaliação de meu trabalho e pelas valiosas colaborações e sugestões.

Ao meu colega de turma Daniel Santos pelo apoio e sessões de estudos em grupo nas disciplinas iniciais do programa. Sem sua ajuda, certamente não teria chegado até aqui.

Ao Prof. Dr. Senthil Kumar Jagatheesaperuma pela oportunidade concedida em trabalhar em conjunto na escrita de um artigo que me trouxe bastante conhecimento científico.

“No futuro, todas as decisões que a humanidade tomar serão informadas por um sistema cognitivo como o Watson, e as nossas vidas serão melhores por causa disso.” (Ginni Rometty, 2015.)

RESUMO

A geração automática de descrições para imagens médicas tem despertado um interesse crescente na área da saúde, devido ao seu potencial para auxiliar profissionais na interpretação e análise de exames clínicos. Neste trabalho, explorou-se o desenvolvimento e avaliação de um modelo generativo de descrições de imagens médicas generalista utilizando o conjunto de dados público *Radiology Objects in COntext* (ROCO). Foi analisado o estado da arte, identificando abordagens baseadas em técnicas de processamento de linguagem natural e aprendizado de máquina. No entanto, foram identificadas algumas lacunas na literatura, como a falta de estudos que explorem o desempenho de modelos específicos para geração de descrições médicas, a necessidade de avaliação objetiva da qualidade das descrições geradas, a falta de generalização dos modelos para diferentes modalidades de imagem e condições médicas, bem como a falta de transparência e interpretabilidade dos modelos. Para abordar esses problemas, foi adotada uma estratégia metodológica que combinou técnicas de processamento de linguagem natural e modelos de reconhecimento de imagens para extrair atributos relevantes das imagens médicas e alimentá-los em um modelo generativo baseado em redes neurais, buscando a generalização dos modelos para diferentes modalidades de imagem e condições médicas. O treinamento do modelo foi realizado utilizando um conjunto de dados anotados e sua avaliação foi feita através das métricas acurácia e *Bilingual Evaluation Understudy* (BLEU). Os resultados obtidos mostraram resultados promissores na geração de descrições de imagens médicas, com uma acurácia de 0,7628 e um BLEU-1 de 0,5387. No entanto, a qualidade das descrições geradas ainda pode ser limitada, apresentando erros semânticos ou falta de detalhes específicos relevantes. Essas limitações podem ser atribuídas à disponibilidade e representatividade dos dados do conjunto de dados ROCO, bem como às técnicas utilizadas para a geração de descrições. Para pesquisas futuras, sugere-se explorar ainda mais a influência de diferentes técnicas e abordagens, como o uso de arquiteturas de redes neurais mais avançadas, implementar a interpretabilidade dos modelos generativos e empregar conjuntos de dados ainda mais amplos e diversificados. Além disso, é recomendada a realização de validação clínica para aprofundar a análise comparativa entre as descrições geradas e as fornecidas por especialistas.

Palavras-chave: Processamento Digital de Imagem; Processamento de Linguagem Natural; Transferência de Conhecimento; Engenharia Biomédica; ROCO.

ABSTRACT

The automatic generation of descriptions for medical images has sparked increasing interest in the healthcare field due to its potential to assist professionals in the interpretation and analysis of clinical exams. This work explores the development and evaluation of a generalist generative model for medical image descriptions using the publicly available Radiology Objects in COntext (ROCO) dataset. The state of the art was analyzed, identifying approaches based on natural language processing techniques and machine learning. However, several gaps were identified in the literature, such as the lack of studies that explore the performance of specific models for medical description generation, the need for objective evaluation of the quality of generated descriptions, the lack of model generalization to different image modalities and medical conditions, as well as the lack of transparency and interpretability of the models. To address these issues, a methodological strategy was adopted, combining natural language processing techniques and image recognition models to extract relevant features from medical images and feed them into a generative model based on neural networks, aiming for model generalization to different image modalities and medical conditions. The model was trained using an annotated dataset, and its evaluation was performed using accuracy and Bilingual Evaluation Understudy (BLEU) metrics. The obtained results showed promising outcomes in the generation of descriptions for medical images, with an accuracy of 0.7628 and a BLEU-1 score of 0.5387. However, the quality of the generated descriptions may still be limited, exhibiting semantic errors or lacking specific relevant details. These limitations can be attributed to the availability and representativeness of the data in the ROCO dataset, as well as the techniques used for description generation. For future research, it is suggested to further explore the influence of different techniques and approaches, such as using more advanced neural network architectures, implementing interpretability of generative models and employing even larger and more diverse datasets. Additionally, conducting clinical validation is recommended to deepen the comparative analysis between generated descriptions and those provided by experts.

Keywords: Digital Image Processing; Natural Language Processing; Transfer Learning; Biomedical Engineering; ROCO.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

BLEU	Avaliação Bilíngue Auxiliar, do inglês 'Bilingual Evaluation Understudy'
CNN	Rede Neural Convolucional, do inglês 'Convolutional Neural Network'
CT	Tomografia Computadorizada, do inglês 'Computed Tomography'
CUI	Identificadores Exclusivos de Conceito, do inglês 'Concept Unique Identifiers'
LSTM	Memória Longa de Curto Prazo, do inglês 'Long Short-Term Memory'
NAS	Busca de Arquitetura Neural, do inglês 'Neural Architecture Search'
NLP	Processamento de Linguagem Natural, do inglês 'Natural Language Processing'
Regex	Expressão Regular
ReLU	Unidade Linear Retificada, do inglês 'Rectified Linear Unit'
RNNs	Redes Neurais Recorrentes, do inglês 'Recurrent Neural Networks'
ROCO	'Radiology Objects in COntext'
SemTypes	Tipos Semânticos, do inglês 'Semantic Types'
UMLS	Sistema Unificado de Linguagem Médica, do inglês 'Unified Medical Language System'
VGG	'Visual Geometry Group'

SUMÁRIO

1 INTRODUÇÃO

A medicina tem sido revolucionada pelo avanço das tecnologias computacionais, que têm desempenhado um papel fundamental no diagnóstico, tratamento e monitoramento de pacientes. Em particular, a aplicação da Inteligência Artificial tem mostrado um potencial promissor para melhorar a qualidade dos cuidados médicos.

A utilização de algoritmos generativos tem se mostrado uma abordagem eficaz para a geração automática de descrições textuais de imagens. Esses algoritmos são capazes de aprender padrões complexos presentes nas imagens e relacioná-los a uma linguagem natural coerente. Dessa forma, eles podem gerar descrições detalhadas e precisas, que refletem as características relevantes presentes nas imagens.

No contexto da medicina, a geração automática de descrições para imagens médicas tem despertado um interesse crescente devido ao seu potencial para auxiliar profissionais na interpretação e análise de exames clínicos (PAVLOPOULOS *et al.*, 2019). As descrições detalhadas e precisas das imagens médicas, como radiografias e ressonâncias magnéticas, desempenham um papel fundamental no diagnóstico e tratamento de pacientes, proporcionando informações valiosas aos profissionais de saúde. No entanto, a produção manual dessas descrições é um processo demorado e suscetível a erros e requer um alto nível de especialização (LYNDON *et al.*, 2017).

Diversas pesquisas têm sido realizadas no campo da geração automática de descrições de imagens, tanto em contextos gerais quanto em aplicações específicas. No entanto, uma parte expressiva desses estudos concentrou-se em imagens de domínios mais amplos, como paisagens, objetos ou animais, deixando uma lacuna em relação às imagens médicas (BEDDIAR *et al.*, 2022a). As particularidades dessas imagens, como a presença de estruturas anatômicas específicas e informações clínicas relevantes, exigem abordagens e técnicas adaptadas ao contexto médico. Portanto, há uma necessidade clara de explorar e desenvolver modelos generativos que sejam capazes de gerar descrições precisas e contextualmente relevantes para as imagens médicas.

O presente trabalho busca abordar o problema de como implementar um modelo generativo de descrições de imagens médicas, considerando a sua aplicação em um conjunto de dados público *Radiology Objects in Context* (ROCO) (PELKA *et al.*, 2018). A questão central que proposta é investigar como utilizar técnicas de processamento de linguagem natural e aprendizado de máquina para gerar descrições precisas e contextualmente relevantes para as

imagens médicas do ROCO.

A abordagem metodológica adotada neste estudo consiste em uma combinação de técnicas de processamento de linguagem natural e aprendizado de máquina. É utilizada uma estratégia onde modelos de reconhecimento de imagens são empregados para extrair os atributos relevantes das imagens médicas. Esses atributos são utilizados como entrada em um modelo generativo baseado em redes neurais que cria as descrições textuais contextualmente relevantes as suas entradas. O treinamento do modelo é realizado utilizando um conjunto de dados anotados, onde cada imagem é associada a uma descrição textual correspondente. A avaliação do modelo é feita através de diversas métricas e comparação com as descrições reais fornecidas.

1.1 Lacunas Identificadas na Literatura

Uma lacuna na literatura está relacionada à falta de estudos que explorem o desempenho de modelos específicos, como MobileNetV2, DenseNet201, ResNet152V2, NASNetLarge, VGG19, Xception, InceptionV3, InceptionResNetV2, em geração de descrições de imagens médicas.

Outra lacuna comum na literatura é a falta de avaliação objetiva de qualidade das descrições geradas. Muitos estudos baseiam-se em métricas automatizadas, como BLEU ou METEOR, que podem não capturar totalmente a relevância clínica e a precisão das descrições médicas. Uma possível contribuição é propor métricas mais adequadas ou abordagens de avaliação que levem em consideração a percepção de especialistas médicos ou avaliações humanas para medir a qualidade das descrições geradas.

Além disso, há uma falta de generalização dos modelos de geração de descrições para diferentes modalidades de imagem e condições médicas. Muitos estudos se concentram em uma única modalidade de imagem ou em condições específicas, limitando a aplicabilidade dos modelos propostos.

Outra lacuna encontrada é a falta de transparência e interpretabilidade dos modelos de geração de descrições. A capacidade de explicar como e por que uma descrição específica é gerada é fundamental, especialmente em aplicações médicas críticas. Uma possível contribuição é explorar abordagens que melhorem a interpretabilidade dos modelos, fornecendo justificativas claras para as descrições geradas e permitindo que os médicos confiem nas informações fornecidas.

1.2 Justificativa

As imagens médicas desempenham um papel crucial no diagnóstico de doenças. Elas fornecem aos médicos informações visuais detalhadas sobre as estruturas e condições internas do corpo humano, permitindo a identificação de anomalias, lesões ou alterações patológicas (SARVAMANGALA; KULKARNI, 2021).

Por exemplo, radiografias, tomografias computadorizadas, ressonâncias magnéticas e ultrassonografias são capazes de identificar alterações no estágio inicial de uma doença, antes mesmo do surgimento de sintomas visíveis ou palpáveis. Isso possibilita a detecção precoce de doenças, aumentando as chances de um tratamento eficaz e de melhores resultados para o paciente. Além disso, auxiliam na identificação e caracterização de lesões, como tumores, fraturas, inflamações e anormalidades congênitas. Elas fornecem informações sobre o tamanho, localização, extensão e características das lesões, permitindo que os médicos determinem a melhor abordagem de tratamento.

Nesse contexto, torna-se relevante o desenvolvimento de modelos generativos capazes de produzir textos descritivos de imagens médicas de forma automatizada e eficiente. Pesquisas anteriores têm explorado diferentes abordagens para a geração automática de descrições, como modelos baseados em regras, modelos estatísticos e modelos baseados em redes neurais (SHARMA; PADHA, 2023). Essas abordagens têm demonstrado resultados promissores em diferentes domínios, como descrição de imagens gerais e descrição de imagens médicas específicas. No entanto, apesar dos avanços recentes nessa área, a geração automática de descrições de imagens médicas ainda apresenta desafios significativos, devido à complexidade e à diversidade dessas imagens médicas.

A geração automática de descrições de imagens médicas tem o potencial de otimizar o fluxo de trabalho dos profissionais de saúde, permitindo uma análise mais eficiente e precisa dos exames clínicos. A aplicabilidade prática dessas soluções pode resultar em benefícios significativos para pacientes e profissionais de saúde, agilizando o processo de tomada de decisão e melhorando a qualidade do atendimento médico (HEILIGER *et al.*, 2022).

1.3 Objetivos

O objetivo geral desta dissertação é desenvolver e avaliar um novo modelo generativo de descrições de imagens médicas generalista, ou seja, que seja capaz de processar diferentes

modalidades de imagem médica e identificar uma variedade de condições clínicas. Para isso, definiu-se os seguintes objetivos específicos:

a) Selecionar imagens e textos para construção de um banco de dados de validação do modelo proposto;
b) Comparar diferentes modelos aprendizado de máquina capazes de gerar descrições textuais a partir das imagens médicas;
c) Investigar a influência de diferentes técnicas, como *transfer learning*, na performance do modelo generativo;
d) Avaliar a qualidade e a adequação das descrições geradas pelo modelo proposto, comparando-as com as descrições reais fornecidas através de matrizes de confusões e suas principais métricas.

1.4 Contribuições do Trabalho

Ao avaliar e comparar o desempenho de modelos específicos em relação à qualidade das descrições geradas, este trabalho contribui para a área de geração de descrições de imagens médicas.

Esta pesquisa contribui explorando a generalização dos modelos para diferentes modalidades de imagem médica, como radiografias, ressonâncias magnéticas ou tomografias computadorizadas e para uma variedade de condições médicas.

Ao abordar algumas das lacunas identificadas na literatura, esta pesquisa fornece avanços significativos teóricos e metodológicos para a área em questão.

1.5 Metodologia Aplicada

A forma clássica de se classificar uma pesquisa se dá por meio de quatro diferentes pilares: Quanto à natureza, quanto à abordagem do problema, quanto aos objetivos e quanto aos procedimentos técnicos (SILVA; MENEZES, 2005).

1.5.1 Natureza da Pesquisa

A pesquisa possui uma natureza aplicada, pois tem como objetivo principal a criação e avaliação de um modelo generativo de descrições de imagens médicas com a finalidade prática de auxiliar profissionais de saúde na interpretação de exames clínicos.

1.5.2 Abordagem do Problema

A abordagem do problema adotada nesta pesquisa é multifacetada, incorporando elementos tanto qualitativos quanto quantitativos. A pesquisa se enquadra como qualitativa devido à exploração aprofundada da geração automática de descrições para imagens médicas e a análise dos resultados obtidos. Ao mesmo tempo, a pesquisa também assume uma abordagem quantitativa ao desenvolver e avaliar os resultados por meio de métricas como acurácia e BLEU-1, permitindo uma análise numérica do desempenho das descrições geradas.

1.5.3 Objetivos da Pesquisa

A pesquisa se caracteriza como exploratória, pois busca preencher lacunas na literatura existente sobre a geração automática de descrições para imagens médicas. Ela também procura identificar desafios e oportunidades na aplicação dessa tecnologia na área da saúde.

1.5.4 Procedimentos Técnicos Adotados

A pesquisa adotou uma abordagem diversificada, incluindo pesquisa bibliográfica para analisar o cenário atual da geração automática de descrições para imagens médicas. Nesse contexto, foram identificadas abordagens já existentes, bem como técnicas de processamento de linguagem natural e aprendizado de máquina aplicadas a esse campo.

Além disso, a pesquisa envolveu aspectos experimentais, onde se conduziram experimentos práticos para o desenvolvimento e treinamento do modelo generativo de descrições de imagens médicas. A base desses experimentos foi formada por técnicas de processamento de linguagem natural, aprendizado de máquina e redes neurais.

Para avaliar a eficácia do modelo gerativo, a pesquisa empregou uma abordagem de pesquisa de avaliação. Isso envolveu a utilização de métricas como acurácia e BLEU-1, permitindo a quantificação objetiva do desempenho das descrições geradas e sua qualidade em comparação com as referências fornecidas.

Além dessas abordagens, o trabalho também pode ser considerado um estudo de caso. A pesquisa concentrou-se na exploração específica da aplicação de geração de descrições para imagens médicas. Com base em análises detalhadas desses casos, foram apresentados resultados e conclusões que ampliam a compreensão do tópico e suas implicações práticas.

1.6 Organização do Trabalho

Esta dissertação está organizada em cinco capítulos. No Capítulo 2, é realizada uma revisão bibliográfica sobre os principais conceitos e técnicas relacionadas à geração automática de descrições de imagens médicas. No Capítulo 3, é descrito em detalhes a metodologia utilizada, incluindo a coleta e preparação dos dados, a implementação do modelo generativo e as métricas de avaliação. No Capítulo 4, são apresentados os resultados obtidos, acompanhados de análises e discussões. Por fim, no Capítulo 5, são apresentadas as conclusões finais do estudo, destacando as contribuições, as limitações e as recomendações para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os principais conceitos teóricos que sustentam a pesquisa desenvolvida neste trabalho, com o intuito de contextualizar a pesquisa proposta neste trabalho, justificar sua relevância no campo da medicina e da inteligência artificial e fornecer uma base sólida de conhecimento, abordando temas como a aplicação de imagens na medicina, o processamento de linguagem natural, o processamento de imagens e a geração de textos para descrições de imagens médicas.

A seguir, na seção 2.1, é abordada a aplicação de imagens na medicina, explorando como as tecnologias de imagem têm sido amplamente utilizadas para auxiliar no diagnóstico, tratamento e monitoramento de doenças. Serão discutidos os principais métodos de aquisição de imagens médicas, como a tomografia computadorizada, a ressonância magnética e a ultrassonografia, destacando sua relevância e contribuições no contexto clínico.

2.1 Aplicação de Imagens na Medicina

A aplicação de imagens na medicina desempenha um papel fundamental no auxílio ao diagnóstico, tratamento e monitoramento de doenças (RAJ *et al.*, 2020). Ao longo dos últimos tempos, as tecnologias de imagem têm evoluído significativamente (GUO *et al.*, 2022), possibilitando a obtenção de informações detalhadas e precisas sobre o corpo humano. Esse fato tem se mostrado essencial para uma melhor compreensão das estruturas anatômicas e fisiológicas do corpo humano, para o estudo de patologias e para o acompanhamento da resposta aos tratamentos.

Dentre as principais modalidades de aquisição de imagens médicas, destacam-se as imagens de Tomografia Computadorizada, do inglês 'Computed Tomography' (CT). A CT é uma técnica que utiliza radiografias de raios X em múltiplos ângulos para obter imagens transversais do corpo humano (WITHERS *et al.*, 2021). Essa modalidade oferece imagens de alta resolução espacial e é amplamente utilizada em diferentes áreas da medicina, como radiologia, cardiologia e oncologia (LOPEZ-MATTEI *et al.*, 2023).

A radiografia, também conhecida simplesmente como raios X, do inglês x-ray, é outra modalidade de imagem comumente utilizada na medicina. Ela envolve a passagem de raios X através do corpo para produzir uma imagem bidimensional dos ossos e estruturas internas (VILLARRAGA-GÓMEZ *et al.*, 2019). As radiografias são amplamente utilizadas para detectar

fraturas, avaliar a integridade óssea e auxiliar no diagnóstico de diversas condições médicas (HARDALAÇ *et al.*, 2022). Também são utilizadas para a avaliação de doenças pulmonares, como pneumonia e tuberculose (ORTIZ-TORO *et al.*, 2022).

Outro tipo de imagem utilizada na medicina é o angiograma, também conhecida como angiografia. Esse procedimento consiste na injeção de um contraste especial para gerar imagens tridimensionais de alta qualidade dos vasos sanguíneos em diferentes partes do corpo para visualizar o fluxo sanguíneo dentro deles, identificando obstruções ou outras anormalidades (METAXAS *et al.*, 2022). A angiografia é especialmente importante no diagnóstico e tratamento de doenças vasculares, como aterosclerose, aneurismas e estenoses, permitindo a seleção adequada das opções de tratamento (ROSTAM-ALILOU *et al.*, 2022).

A CT abdominal é outra aplicação valiosa da tomografia computadorizada, fornecendo imagens detalhadas dos órgãos abdominais, como fígado, rins, pâncreas e baço. Essa modalidade de imagem é amplamente utilizada na detecção e no acompanhamento de doenças abdominais, incluindo cânceres, infecções, cálculos biliares, sangramentos e anormalidades estruturais (SCHWARTZ *et al.*, 2023).

A aplicação dessas tecnologias de imagem tem sido fundamental na prática médica, possibilitando a detecção precoce de patologias, o planejamento de intervenções cirúrgicas, o monitoramento da eficácia de tratamentos e o acompanhamento da evolução de doenças ao longo do tempo, desempenhando um papel essencial na medicina moderna.

Na seção 2.2, será abordada a extração de atributos, uma disciplina interdisciplinar que envolve a manipulação e análise de imagens digitais. Serão explorados os principais conceitos dos algoritmos utilizados no processamento de imagens para a extração de atributos.

2.2 Extração de Atributos

A extração de atributos é uma área de visão computacional que envolve a manipulação e análise de imagens digitais e envolve uma série de técnicas e algoritmos para aprimorar, segmentar, extrair informações e interpretar dados visuais contidos em imagens (BURGER; BURGE, 2016). No contexto médico, a extração de atributos de imagens é amplamente utilizado para auxiliar na análise de exames e diagnósticos, permitindo a geração de informações relevantes que aprimoram a tomada de decisão clínica (KORA *et al.*, 2022).

Existem várias abordagens para extrair atributos de imagens digitais, como a Transformada de Fourier que permite analisar as frequências presentes na imagem e identificar padrões

específicos (ZORIĆ *et al.*, 2022). Outra abordagem popular é a extração de características baseada em texturas, que envolve a análise de padrões de textura presentes nas imagens, utilizando algoritmos como Matriz de Coocorrência de Níveis de Cinza (GLCM) ou Transformada de Wavelet (AGGARWAL, 2022). Também é possível utilizar abordagens baseadas em aprendizado de máquina, como as Rede Neural Convolucional, do inglês 'Convolutional Neural Network' (CNN).

Dentre essas diferentes abordagens, destaca-se a CNN que é um modelo de aprendizado de máquina especialmente projetados para processar dados de forma hierárquica, capturando informações visuais relevantes em diferentes níveis de abstração. As CNN*s* são frequentemente usadas para extrair características visuais de imagens médicas (HU *et al.*, 2023). Nas próximas subseções, será feita uma breve apresentação das CNN*s* utilizadas neste trabalho.

2.2.1 MobileNetV2

O MobileNetV2 é uma arquitetura de CNN projetada para tarefas de visão computacional com restrições de recursos, como dispositivos móveis ou sistemas embarcados. Ele usa uma combinação de convoluções separáveis em profundidade e convoluções regulares para obter um equilíbrio entre eficiência computacional e desempenho (SANDLER *et al.*, 2018). No trabalho de TOĞAÇAR *et al.* (2021), foi proposto um modelo de detecção inteligente de câncer de pele aplicando MobileNetV2.

2.2.2 DenseNet201

O DenseNet201 é uma arquitetura de CNN que se destaca por sua conectividade densa entre suas camadas. Ao contrário das CNN*s* tradicionais, que conectam camadas sequencialmente, o DenseNet201 conecta todas as camadas subsequentes entre si, permitindo o compartilhamento direto de informações entre elas. Essa conectividade densa ajuda a mitigar o problema do desvanecimento do gradiente e permite um melhor fluxo de informações através da rede (HUANG *et al.*, 2016). No trabalho de JAISWAL *et al.* (2020), esta arquitetura foi utilizada na classificação de pacientes infectados com COVID-19 por meio de seus exames radiológicos.

2.2.3 ResNet152V2

A ResNet152V2 é uma versão aprofundada da arquitetura ResNet que se baseia no conceito de resíduos. Essa arquitetura utiliza conexões de atalho que saltam camadas, permitindo o fluxo direto de informações através da rede. Essa abordagem ajuda a superar o problema do desvanecimento do gradiente em redes mais profundas e facilita o treinamento de CNN*s* profundas (HE *et al.*, 2016). No trabalho de KAKKAR *et al.* (2023), utilizou-se esta arquitetura para a detecção e classificação de diferentes classes de parasitas da malária usando imagens microscópicas.

2.2.4 NASNetLarge

O NASNetLarge é uma arquitetura de CNN projetada por meio de uma pesquisa automatizada de arquiteturas de redes neurais chamada Busca de Arquitetura Neural, do inglês 'Neural Architecture Search' (NAS). Essa abordagem automatizada permite que o modelo seja adaptado para atender às necessidades específicas da tarefa a qual ele foi treinado (ZOPH *et al.*, 2017). No trabalho de RAMESH *et al.* (2023), esta arquitetura foi utilizada na detecção de retinopatia diabética em imagens do fundo da retina com otimização bioinspirada.

2.2.5 VGG19

A arquitetura VGG19 é um modelo de CNN amplamente reconhecido e utilizado em tarefas de visão computacional. Desenvolvida pela 'Visual Geometry Group' (VGG) na Universidade de Oxford, a VGG19 é uma extensão da arquitetura VGG16, composta por 19 camadas de convolução e camadas totalmente conectadas. Essa arquitetura se destaca por sua simplicidade e profundidade, utilizando filtros de tamanho 3x3 e camadas de pooling para extrair características de imagens em diferentes níveis de abstração (SIMONYAN; ZISSERMAN, 2014). A VGG19 alcançou excelente desempenho em desafios de classificação de imagens, como o *ImageNet*, e seu design modular tornou-se uma referência na área, servindo como base para o desenvolvimento de novos modelos de redes neurais convolucionais. No trabalho de ALKHUZAIE *et al.* (2023), utilizou-se esta arquitetura para detectar leucemia linfoblástica aguda.

2.2.6 Xception

O Xception é uma arquitetura de CNN que se baseia no conceito de separação de canais espaciais e de cor. Ele substitui as operações convolucionais tradicionais por camadas separáveis em profundidade, separando a convolução em duas etapas: uma convolução espacial e uma convolução de canal (CHOLLET, 2016). Isso resulta em um modelo mais eficiente e com maior poder de representação. No trabalho de ULLAH *et al.* (2022), utilizou-se esta arquitetura para detectar e identificar tumores cerebrais a partir de imagens de ressonância magnética.

2.2.7 InceptionV3

A arquitetura InceptionV3 é um poderoso modelo de rede neural convolucional desenvolvido pelo Google. É uma evolução das versões anteriores da arquitetura Inception, projetada para lidar com o desafio de equilibrar a profundidade da rede com a eficiência computacional. A InceptionV3 utiliza um módulo de convolução denominado 'Inception' que combina filtros de diferentes tamanhos de *kernel* em paralelo para capturar informações em diferentes escalas (SZEGEDY *et al.*, 2016b). Essa abordagem permite que a rede extraia recursos ricos e complexos das imagens. Além disso, a arquitetura emprega técnicas como regularização e normalização para mitigar problemas como *overfitting*. No trabalho de FAIZAL *et al.* (2023), esta arquitetura foi utilizada na detecção automatizada de catarata em imagens oculares.

2.2.8 InceptionResNetV2

A arquitetura InceptionResNetV2 é uma poderosa fusão entre as arquiteturas Inception e ResNet, desenvolvida pelo Google. Ela combina a capacidade de extração de características multiescala da Inception com a resiliência ao treinamento profundo da ResNet (SZEGEDY *et al.*, 2016a). A InceptionResNetV2 emprega blocos residuais que permitem o fluxo de informações diretas entre camadas, facilitando o treinamento de redes mais profundas. Além disso, a arquitetura utiliza módulos Inception que capturam informações contextuais em diferentes escalas, enriquecendo a representação dos dados. Com isso, a InceptionResNetV2 alcançou resultados notáveis em várias tarefas de visão computacional, como classificação de imagens, detecção de objetos e segmentação semântica. Sua combinação única de características das duas arquiteturas tornou-a uma opção popular para aplicações exigentes em que uma representação rica e uma resiliência ao treinamento profundo são essenciais. No trabalho de MEEM *et al.*

(2023), utilizou-se esta arquitetura na detecção de células de citomorfologia na medula óssea usando.

Em seguida, na seção 2.3, será abordado o processamento de linguagem natural, uma área da inteligência artificial que visa capacitar sistemas computacionais a entender, interpretar e gerar linguagem humana de forma automatizada. Serão apresentadas as principais técnicas e algoritmos utilizados nessa área, bem como suas aplicações em diversas áreas, incluindo a medicina.

2.3 Processamento de Linguagem Natural

O Processamento de Linguagem Natural, do inglês 'Natural Language Processing' (NLP) é uma área de pesquisa que combina conhecimentos de linguística, ciência da computação e inteligência artificial para permitir que as máquinas compreendam, interpretem e gerem linguagem humana de forma semelhante aos seres humanos (TUNSTALL *et al.*, 2022). O NLP tem encontrado diversas aplicações em várias áreas, incluindo a análise e geração de descrições de imagens médicas, onde a geração automatizada de textos descritivos pode trazer benefícios significativos no diagnóstico dos pacientes (LIU *et al.*, 2022).

Para a geração de textos, diferentes abordagens podem ser aplicadas, como modelos de linguagem baseados em Redes Neurais Recorrentes, do inglês 'Recurrent Neural Networks' (RNNs) ou CNN*s* combinadas com mecanismos de atenção (KHURANA *et al.*, 2022). Essas abordagens podem ser treinadas usando algoritmos de aprendizado supervisionado, onde pares de imagens e descrições são usados para aprender a mapear características visuais em sequências de palavras descritivas (AYESHA *et al.*, 2021).

Dentre essas abordagens, destacam-se os Transformadores, que são modelos de aprendizado de máquina baseados em mecanismos de atenção. Em vez de processar sequencialmente as palavras como as RNNs, os Transformadores calculam as interações entre todas as palavras em uma frase usando operações de atenção (VASWANI *et al.*, 2017). Esses modelos têm sido aplicados com sucesso na geração de descrições de imagens médicas, pois têm a capacidade de capturar relacionamentos complexos entre palavras e gerar descrições mais contextualmente relevantes (HE *et al.*, 2023).

A avaliação da qualidade das descrições médicas geradas é um aspecto crucial. Existem várias métricas de avaliação, como Acurácia, BLEU, METEOR, CIDEr e ROUGE que podem ser usadas para comparar as descrições geradas com as descrições de referência

fornecidas pelas as anotações das imagens (WU *et al.*, 2023).

Inicialmente desenvolvido para avaliar a qualidade de traduções, a Avaliação Bilíngue Auxiliar, do inglês 'Bilingual Evaluation Understudy' (BLEU), tornou-se uma métrica amplamente empregada na área de processamento de linguagem natural, não apenas para traduções, mas também para a geração de textos em geral. Essa métrica mensura a similaridade entre um texto gerado por uma máquina e um texto de referência humano, levando em conta a sobreposição de palavras e a ordem das frases. A BLEU compara n-gramas (sequências de n palavras) da descrição gerada com os da descrição de referência e atribui uma pontuação de 0 a 1 ao texto gerado, sendo 1 a correspondência perfeita (PAPINENI *et al.*, 2001).

Portanto, a utilização da métrica BLEU neste trabalho oferece um contexto valioso para situar o desempenho do modelo desenvolvido em relação ao estado da arte, contribuindo para uma análise mais abrangente dos resultados obtidos.

A seguir, na seção 2.4, será abordado os otimizadores de gradientes, os quais desempenham um papel crucial no contexto do aprendizado de máquina.

2.4 Otimizadores de Gradiente

Esta seção aborda os otimizadores de gradiente utilizados neste trabalho. Os otimizadores de gradiente são essenciais na etapa de treinamento dos modelos, pois determinam como os parâmetros do modelo são atualizados durante o processo de otimização (LU, 2022).

Nesse contexto, 4 otimizadores serão abordados: Adam, AdamW, Adadelta e Adafactor. Para cada um deles, será apresentada uma breve descrição e sua aplicabilidade na geração de textos descritivos de imagens médicas. Essa análise é fundamental para compreender como esses otimizadores podem melhorar a qualidade da geração de descrições precisas e coerentes.

2.4.1 Adam

O otimizador Adam é um dos mais populares em aprendizado de máquina que combina o poder do método de gradiente descendente estocástico com o conceito de adaptação de taxa de aprendizado (KINGMA; BA, 2014). Ele calcula taxas de aprendizado adaptativas individuais para cada parâmetro, com base em estimativas do primeiro momento (média) e segundo momento (variância) dos gradientes. Essa abordagem permite que o algoritmo ajuste as taxas de aprendizado de forma dinâmica, otimizando a convergência do modelo. No contexto da

geração de textos descritivos de imagens médicas, o otimizador Adam pode ser útil para ajustar os parâmetros do modelo, permitindo uma busca eficiente no espaço de aprendizado.

2.4.2 AdamW

O otimizador AdamW é uma extensão do algoritmo Adam, que introduz uma correção para lidar com a regularização L2, também conhecida como decaimento de peso. A adição o termo de regularização L2 ajuda a evitar o *overfitting*, tornando o modelo mais robusto. O AdamW atualiza os pesos do modelo considerando tanto a média do gradiente quanto a variância, além de aplicar a regularização L2 (LOSHCHILOV; HUTTER, 2017). Essa combinação permite uma otimização mais estável e geralmente resulta em melhor desempenho. Na geração de textos descritivos de imagens médicas, o AdamW pode ser uma opção interessante para ajustar os pesos do modelo, controlando o *overfitting* e melhorando a qualidade das descrições geradas.

2.4.3 Adadelta

O otimizador Adadelta é uma técnica adaptativa que visa resolver o problema de escolha adequada da taxa de aprendizado no método de gradiente descendente. Ele utiliza uma estratégia de adaptação local baseada na média móvel ponderada dos gradientes recentes para ajustar a taxa de aprendizado de forma adaptativa (ZEILER, 2012). A principal vantagem do Adadelta é que ele não requer a especificação de uma taxa de aprendizado inicial, reduzindo assim a necessidade de ajuste manual. No contexto da geração de textos descritivos de imagens médicas, o Adadelta pode ajudar a ajustar as taxas de aprendizado de forma eficaz, melhorando a convergência e a estabilidade do modelo durante o treinamento.

2.4.4 Adafactor

O otimizador Adafactor é uma técnica recente que combina características de adaptação de taxa de aprendizado do Adam com a eficiência de computação do método de otimização diagonal de Gauss-Newton (SHAZEER; STERN, 2018). Ele atualiza os parâmetros do modelo usando um fator de aprendizado adaptativo, que é calculado com base na matriz de covariância dos gradientes. Ao contrário de outros otimizadores que utilizam taxa de aprendizado escalar, o Adafactor ajusta individualmente as taxas de aprendizado para cada parâmetro, melhorando a eficiência computacional. Na tarefa de geração de textos descritivos de imagens médicas,

o Adafactor pode oferecer um compromisso entre eficiência e adaptabilidade, permitindo um treinamento mais rápido e uma melhor otimização do modelo

A seguir, na seção 2.5, serão abordados brevemente os conceitos de *Transfer Learning* e *Image Augmentation*, técnicas complementares que foram utilizadas neste trabalho.

2.5 Técnicas Adicionais

Nesta seção, serão abordadas as técnicas de *Transfer Learning* e *Image Augmentation* que foram utilizadas neste trabalho como estratégias para melhorar a eficiência e a capacidade de generalização dos modelos. Serão explorados os conceitos fundamentais do *Transfer Learning*, incluindo a utilização de modelos pré-treinados e a adaptação desses modelos para tarefas específicas, além do conceito de *Image Augmentation*, utilizada para aumentar a diversidade do conjunto de dados de treinamento (XU *et al.*, 2023).

2.5.1 Transfer Learning

O *Transfer Learning* é uma técnica em que um modelo pré-treinado em um grande conjunto de dados de imagens, como o *ImageNet* (DENG *et al.*, 2009), é utilizado como ponto de partida para tarefas específicas (KIM *et al.*, 2022). O modelo pré-treinado já aprendeu a extrair recursos de imagens gerais e, em seguida, pode ser ajustado em um conjunto de dados menor e específico para sua tarefa, como imagens médicas.

O *Transfer Learning* do *ImageNet* é uma abordagem comum para superar a escassez de imagens médicas anotadas. Ao utilizar modelos pré-treinados em um grande conjunto de dados, como o *ImageNet*, é possível aproveitar o conhecimento adquirido por esses modelos sobre características visuais gerais em outro contexto, como o de imagens médicas (MATSOUKAS *et al.*, 2022). Com esse ponto de partida, é possível ajustar esses modelos em um conjunto de dados menor e específico. A expectativa de se utilizar esta técnica neste trabalho é melhorar a capacidade de geração de descrições das imagens médicas.

2.5.2 Image Augmentation

O *Image Augmentation* é uma técnica em que imagens de treinamento são modificadas de forma aleatória e controlada, através da aplicação de rotações, translações, redimensionamentos, reflexões e outras transformações, gerando novas imagens com pequenas variações (XU

et al., 2023), aumentando a quantidade e a diversidade dos dados de treinamento. No contexto médico, o uso dessa técnica resultou em um aumento da capacidade dos modelos de generalizar para dados não vistos (CHEN *et al.*, 2022).

Aumentar a quantidade de dados de treinamento é essencial para lidar com algumas das dificuldades inerentes do contexto médico, como a alta diversidade dos diferentes casos clínicos possíveis e o número reduzido de imagens médicas disponíveis (KHALIFA *et al.*, 2021). A expectativa de se utilizar esta técnica neste trabalho é aprimorar a capacidade de geração de descrições em diferentes contextos clínicos.

Por fim, na seção 2.6, é explorada a aplicação da geração de textos para descrições de imagens médicas. Serão discutidos os desafios e oportunidades específicos nessa área, bem como as abordagens e metodologias adotadas para gerar descrições automáticas de imagens médicas. Serão apresentados pesquisas relevantes que demonstram a eficácia e o potencial dessa aplicação.

2.6 Revisão da Literatura

Esta seção tem como objetivo analisar alguns modelos de geração de descrições de imagens médicas presentes na literatura. Nesta seção, serão apresentados e discutidos alguns estudos relevantes da área que foram selecionados com base nos critérios de seleção apresentados na próxima subseção.

2.6.1 Critérios de Seleção de Trabalho na Literatura

Primeiramente, para garantir que os artigos selecionados fossem relacionados ao tema da pesquisa, utilizou-se as seguintes palavras-chave:

a) Modelo generativo;
b) Descrições textuais;
c) Imagens médicas;
d) Processamento de linguagem natural;
e) Processamento de imagens.

Com essas palavras-chave, gerou-se diversas buscas sistemática nas principais bases de dados acadêmicas, como IEEE Xplore, ScienceDirect, PubMed, arXiv e Google Scholar.

O ano de publicação foi um critério de eliminação. Definiu-se que seriam analisados

apenas trabalhos publicados de 2020 em diante, para que a análise da literatura refletisse os avanços mais recentes neste campo de pesquisa.

Além disso, avaliou-se a metodologia utilizada nos estudos, selecionando aqueles que apresentaram uma abordagem clara e detalhada sobre como o modelo generativo foi implementado, bem como as métricas de avaliação utilizadas.

Outro critério utilizado diz respeito aos resultados experimentais. Selecionou-se apenas aqueles que tinham resultados consistentes e respaldados por uma discussão aprofundada que comparavam com outras abordagens existentes.

2.6.2 Atual Estado da Arte

Na Tabela 1, é resumido os estudos selecionados com os critérios apresentados anteriormente. Essa tabela contém as seguintes informações:

a) Extração de Atributos: Indica os modelos de extração de atributos utilizados para processar as imagens antes da geração das descrições;
b) Geração de Descrições: Descreve as abordagens utilizadas para a geração de descrições das imagens médicas;
c) Otimizador Gradiente: Indica os otimizadores de gradiente utilizados durante o treinamento dos modelos;
d) Atenção: Indica se a técnica de atenção foi empregada nos modelos. A atenção é uma abordagem que permite que o modelo foque em partes relevantes da imagem ao gerar as descrições;
e) *Transfer Learning* (TL): Indica se os modelos utilizaram transferência de aprendizado, aproveitando o conhecimento prévio de modelos pré-treinados para melhorar o desempenho na geração de descrições;
f) Treinável (TR): Indica se os pesos dos modelos foram ajustáveis durante a etapa de treinamento;
g) *Image Augmentation* (IA): Indica se a técnica de aumento de dados foi aplicada durante o treinamento dos modelos;
h) BLEU-1: Indica as pontuações obtidas na métrica de avaliação BLEU-1, que mensura a sobreposição de palavras entre as descrições geradas e as descrições de referência;
i) Conjunto de Dados: Descreve os conjuntos de dados utilizados nos estudos;

j) Entrada: Indica quais são as entradas para os modelos;

k) Saída: Descreve o tipo de descrição gerada pelos modelos.

Tabela 1 – Modelos de Geração de Descrições de Imagens Médicas

Referência	Extração de Atributos	Geração de Descrições	Otimizador Gradiente	Atenção	TL	TR	IA	BLEU-1	Conjunto de Dados	Entrada	Saída
(PARK *et al.*, 2020)	ResNet152	LSTM	Adam	✓	✓	✓		0,3293	IU X-RAY	Raios X	Descrições detalhadas de raios X do tórax
(HUANG *et al.*, 2021)	DNN Customizada	LSTM	SGD		✓	✓		0,2190	DeepEyeNet	Imagens de retinas e palavras-chave	Descrições detalhadas de imagens de retinas
(PARK *et al.*, 2021)	ResNet152	LSTM e Transformador	Adam	✓	✓	✓		0,3731	PEIR GROSS e ICLEF-CAPTION	Raios X	Descrições detalhadas de raios X do tórax
(SINGH *et al.*, 2022)	CNN Customizada	LSTM	?	✓				?	VQA-Med	Imagens Diversas	Uma única palavra entre 7 possíveis
(BEDDIAR *et al.*, 2022b)	MLC+VGG-16	LSTM	?	✓	✓	✓	✓	?	ImageCLEFmed	Imagens diversas e palavras-chave	Descrições detalhadas de imagens diversas
(SONG *et al.*, 2022)	ResNet101	LSTM	Adam	✓	✓	✓		0,5230 e 0,5700	Open-i e LGK	Raios X e Ultrasons	Descrições simples, de 1 palavra
(LEE *et al.*, 2022)	CNN Customizada + ROI	Transformador	Adam	✓	✓	✓		0,5064	IU X-RAY	Raios X	Descrições detalhadas de raios X do tórax
(SINGH *et al.*, 2023)	VGG16 ResNet152V2 DenseNet201	Gated Recurrent Unit	?	✓	✓	✓		0,5160	Open-i	Raios X	Descrições detalhadas de raios X do tórax
Este Trabalho	MobileNetV2 DenseNet201 ResNet152V2 NASNetLarge VGG19 Xception InceptionV3 Inception-ResNetV2	Transformador	Adam AdamW Adadelta Adafactor	✓	✓	✓	✓	0,5387	ROCO	Imagens diversas do corpo	Descrições simples de diferentes modalidades de imagem e condições médicas

Fonte: Elaboração própria.

Em PARK *et al.* (2020), é proposto o uso de um modelo de memória de longo prazo (LSTM) combinado com diferenças de características e informações de marcação para gerar relatórios de radiografias de tórax automaticamente. O objetivo é reduzir a carga de trabalho dos médicos e economizar tempo e despesas. O estudo sugere dois modelos: um modelo DiTag que utiliza diferenças entre imagens de pacientes e imagens normais, e um modelo mDiTag que também considera diferenças de baixo nível, como contraste, textura e área localizada. A avaliação mostra que o modelo mDiTag supera os outros modelos, fornecendo mais informações para a geração de legendas.

Em HUANG *et al.* (2021), é desenvolvido um novo modelo de rede neural para a geração automática de relatórios médicos para imagens da retina. O modelo utiliza informações contextuais e multimodais para criar relatórios mais precisos e significativos. Os resultados experimentais mostram que o método proposto supera os modelos de referência, alcançando um desempenho superior em métricas comuns de avaliação. Isso indica que o modelo é capaz de aproveitar as informações interativas entre a imagem de entrada e o contexto, resultando em relatórios de maior qualidade.

Em PARK *et al.* (2021), é proposto o desenvolvimento de um modelo de legenda de imagens para radiografias de tórax, com o objetivo de reduzir a carga de trabalho dos médicos na escrita de relatórios. O modelo considera as diferenças entre imagens de pacientes e imagens normais, utilizando uma memória de longo prazo hierárquica (LSTM) ou um transformador

como decodificador. Foram realizados experimentos com diferentes métodos de representação de características e avaliados usando métricas como BLEU, METEOR, ROUGE-L e CIDEr. O modelo mais eficaz foi o que utilizou um transformer com múltiplas diferenças, sem agrupamento médio global e que aplicou o produto elemento a elemento. Concluiu-se que o decodificador transformador é mais adequado do que o LSTM hierárquico e o produto elemento a elemento é mais eficaz do que a subtração para expressar as diferenças de características.

Em SINGH *et al.* (2022), é proposto um novo modelo chamado *"show, attend, and tell"* (ATM) que utiliza um mecanismo de atenção visual em um modelo codificador-decodificador. Para otimizar os parâmetros iniciais do ATM, é aplicado um algoritmo evolucionário chamado *Strength Pareto Evolutionary Algorithm-II (SPEA-II)*. O desempenho do modelo é avaliado em conjuntos de dados de referência e comparado com outras técnicas de legenda de imagens médicas. Os resultados mostram que o ATM baseado em SPEA-II supera significativamente os modelos existentes em termos de desempenho.

Em BEDDIAR *et al.* (2022b), foi proposto o uso de conceitos médicos associados às imagens para gerar descrições mais adequadas. O modelo proposto é composto por um codificador de características semânticas, um codificador de características visuais e um modelo LSTM para geração de texto. A busca em feixe é utilizada para selecionar as melhores palavras seguintes com base nas características da imagem médica. A proposta é avaliada em um conjunto de dados médicos e os resultados mostram a eficácia e eficiência do método desenvolvido.

Em SONG *et al.* (2022), propõe-se um método que utiliza redes generativas adversárias condicionais e um avaliador de estilo de linguagem. O sistema consiste em três módulos: gerador de relatório de diagnóstico, discriminador e avaliador de estilo de linguagem. O gerador de relatório utiliza mecanismos de atenção e extração de características para produzir relatórios adequados para o contexto clínico. Um discriminador baseado em RNN é utilizado para verificar a autenticidade dos relatórios gerados, e um avaliador de estilo de linguagem garante a consistência de estilo entre os relatórios gerados e os reais. Além disso, o uso de aprendizado por reforço ajuda a produzir relatórios de diagnóstico de alta qualidade e a superar problemas de discretização no treinamento adversário. Os experimentos mostram que o método proposto apresenta melhorias significativas em relação a outros métodos de geração de descrições de imagens médicas, tanto em avaliações objetivas quanto subjetivas.

Em LEE *et al.* (2022), propõe-se um método usando abordagens baseadas em transformadores. O objetivo é superar a limitação das abordagens atuais, que geram texto apenas

a partir das características globais de uma imagem inteira. Para isso, são propostos dois métodos principais: o Extrator Visual Global-Local (GLVE) e o Codificador-Decodificador Cruzado do Transformer (CEDT). O GLVE captura tanto características globais quanto características locais da imagem, permitindo a geração de legendas mais detalhadas. O CEDT injeta múltiplas camadas de características do codificador no processo de decodificação, possibilitando uma descrição mais completa das características gerais da imagem. Os métodos propostos contribuem para melhorias no desempenho do modelo de geração de legendas, permitindo a descrição de elementos como o tamanho de órgãos e a estrutura óssea. Os experimentos realizados mostraram que o modelo proposto superou os resultados de referência baseados em transformadores, apresentando um aumento significativo nas métricas de avaliação, como pontuação BLEU, METEOR e ROUGE-L.

Em SINGH *et al.* (2023), é proposta uma rede chamada DCNet, que combina três modelos pré-treinados (VGG16, ResNet152V2 e DenseNet201) para obter melhores resultados. No entanto, a DCNet é sensível aos seus parâmetros de controle e por isso é proposto um método chamado EDC-Net para ajustar esses parâmetros usando um processo de evolução diferencial. Os experimentos mostram que o EDC-Net é capaz de extrair eficientemente as características relevantes das imagens médicas. Em comparação com modelos existentes, o EDC-Net apresenta um desempenho superior em métricas como BLEU-1, BLEU-2, BLEU-3, BLEU-4 e estatísticas kappa (KS) no conjunto de dados Open-i.

Analisando a literatura, verificou-se que muitos estudos têm explorado o uso de abordagens de aprendizado de máquina, como CNN*s* e RNNs, para a geração de descrições de imagens médicas. As CNN*s* são usadas para extrair características visuais relevantes das imagens, enquanto as RNNs, como a Memória Longa de Curto Prazo, do inglês 'Long Short-Term Memory' (LSTM) e os Transformadores são usadas para gerar as descrições sequenciais.

Além disso, percebeu-se que sempre foi adotando uma abordagem supervisionada, em que modelos são treinados com conjuntos de dados anotados, com imagens médicas combinadas com suas respectivas descrições.

Ainda analisando a literatura, percebeu-se que foi explorado o uso de mecanismos de atenção para melhorar a qualidade das descrições geradas. Esses mecanismos permitem que o modelo se concentre em partes específicas da imagem ao gerar cada palavra da descrição, o que pode levar a descrições mais relevantes e contextualmente corretas.

Percebeu-se que abordagens de transferência de conhecimento têm sido aplicadas, em que modelos pré-treinados em tarefas relacionadas, como reconhecimento de objetos em imagens

naturais, são finamente ajustados para a tarefa de geração de descrições de imagens médicas. Isso pode ajudar a melhorar o desempenho quando há falta de dados anotados específicos para a tarefa em questão.

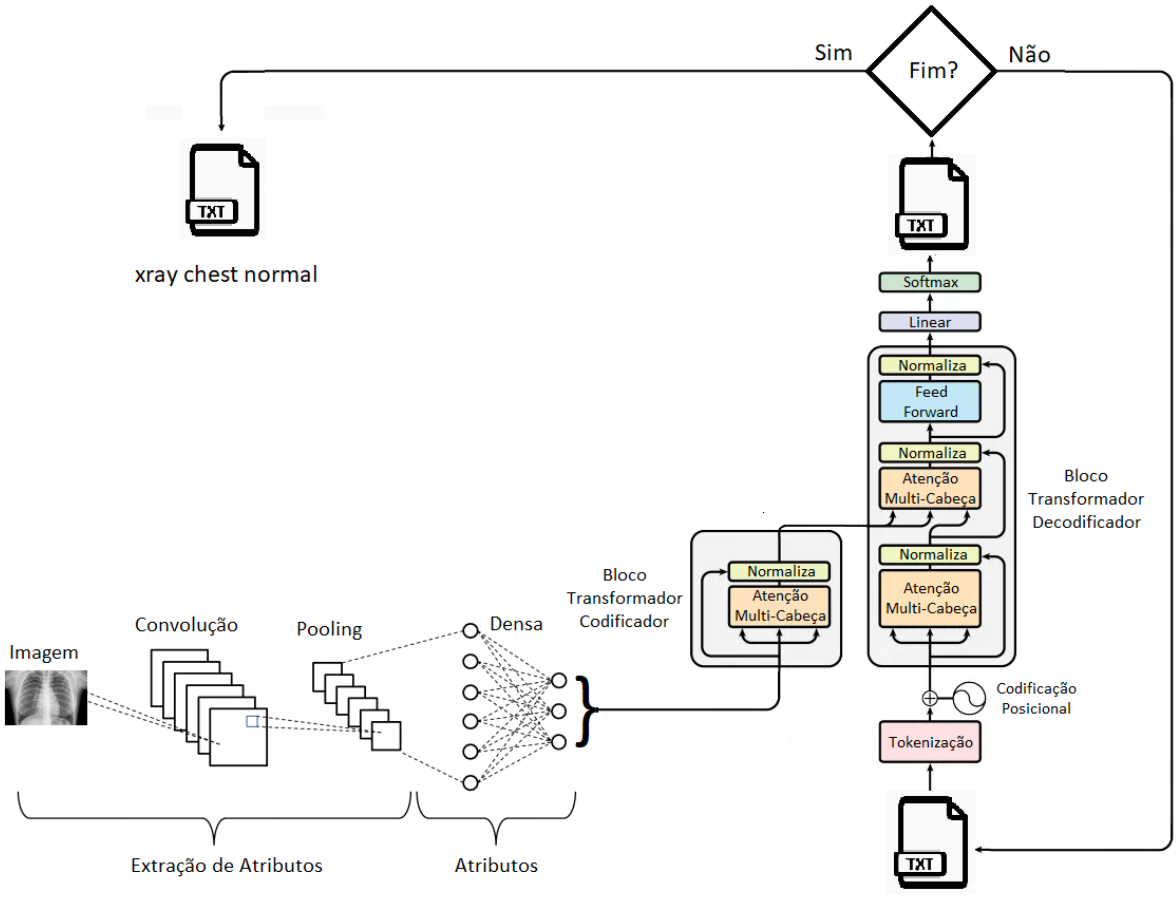
A criação e anotação de conjuntos de dados grandes e diversificados para treinar modelos de geração de descrições de imagens médicas é um desafio. A falta de conjuntos de dados suficientes e abrangentes dificulta a validação e a comparação adequada dos métodos propostos. Por isso, é comum encontrar na literatura estudos que apliquem técnicas de *Image Augmentation*.

3 METODOLOGIA

Neste capítulo, são apresentadas todas as técnicas computacionais utilizadas para se gerar descrições textuais de imagens médicas. Primeiramente, a base de dados ROCO é apresentada. Em seguida, é descrito o algoritmo de seleção de um subconjunto de imagens e textos dessa base de dados. Logo depois, é explicado em detalhes como o experimento para investigar as combinações possíveis das variáveis estudadas foi realizado, a fim de se obter a melhor combinação das técnicas analisadas. Subsequentemente, é apresentado algoritmo generativo construído utilizando técnicas de *deep learning* e linguagem natural para a geração de descrições de imagens médicas. Por fim, são descritos os parâmetros utilizados na avaliação de desempenho do algoritmo generativo e os recursos computacionais utilizados.

Na Figura 1, é representado um esquema resumido do modelo generativo de descrições médicas proposto. Nas próximas seções, cada um dos elementos nesse infográfico será explorado. Entretanto, o primeiro passo é entender o conjunto de dados utilizado.

Figura 1 – Infográfico do Modelo Proposto

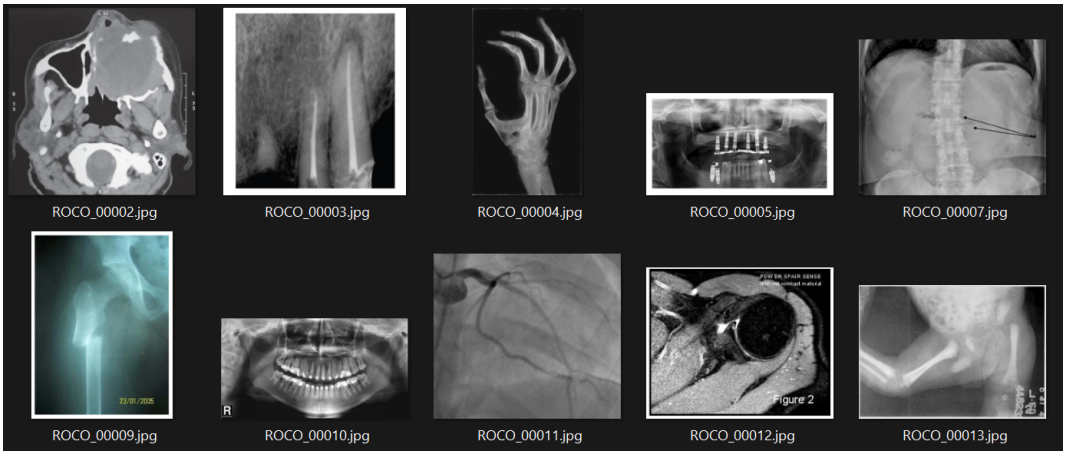


Fonte: Elaboração própria.

3.1 Descrição da Base ROCO

Neste trabalho, utilizou-se o conjunto de dados *Radiology Objects in Context* (ROCO). Esse conjunto consiste de aproximadamente 82 mil imagens de radiologia de várias modalidades diferentes, incluindo tomografia computadorizada, ultrassom, raios X, fluoroscopia, tomografia por emissão de pósitrons, mamografia, ressonância magnética, angiografia e outros. Na Figura 2, é ilustrado um subconjunto exemplo dessas imagens.

Figura 2 – Imagens pertencem ao subconjunto ‘Radiologia’ do conjunto de dados ROCO



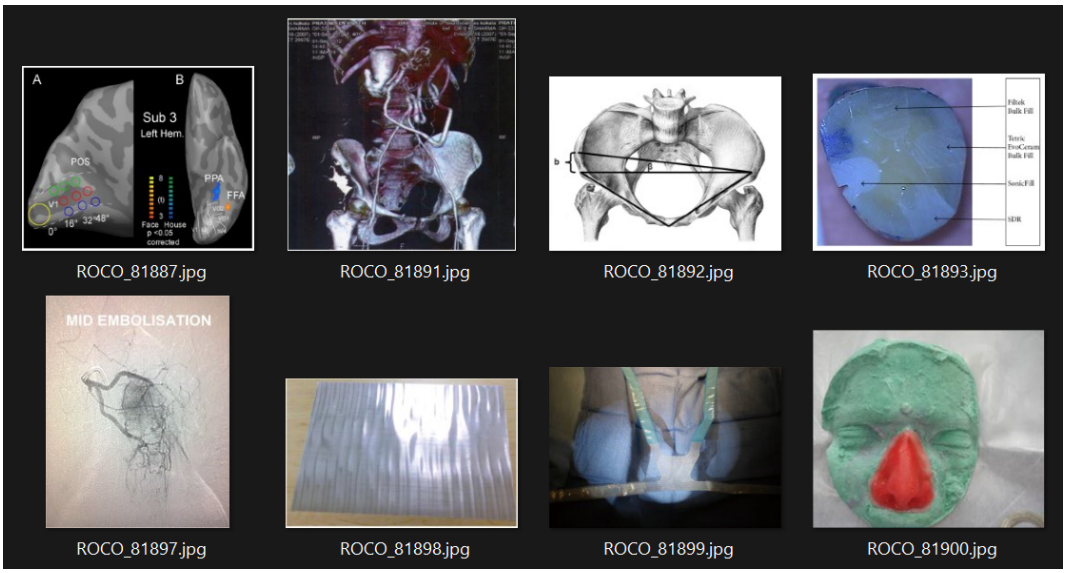
Fonte: Elaboração própria.

Esse conjunto de dados foi construído e disponibilizado na pesquisa de Pelka *et al.* (2018). Todas as imagens do ROCO são de publicações disponíveis no *PubMed Central Open Access*. Segundo os autores, o ROCO pode ser usado para construir modelos generativos para descrições de imagens, modelos de classificação para categorização e marcação de imagens ou sistemas de recuperação de imagens baseados em conteúdo.

Além das imagens radiológicas, é fornecido um conjunto de imagens médicas de outras naturezas, tal como figuras, retratos, artes digitais, ilustrações e fotos clínicas. Esse conjunto pode ser utilizado para melhorar o desempenho de previsões e classificações, entretanto não foi utilizado neste trabalho. Na Figura 3, é ilustrado um subconjunto qualquer de exemplo dessas imagens.

Todas as imagens presentes no conjunto de dados ROCO possuem diversos arquivos de texto contendo uma descrição correspondente da imagem, palavras-chave que a caracterizam, dois Sistema Unificado de Linguagem Médica, do inglês ‘Unified Medical Language System’ (UMLS) - Identificadores Exclusivos de Conceito, do inglês ‘Concept Unique Identifiers’ (CUI), Tipos Semânticos, do inglês ‘Semantic Types’ (SemTypes) - e os respectivos links para *download*.

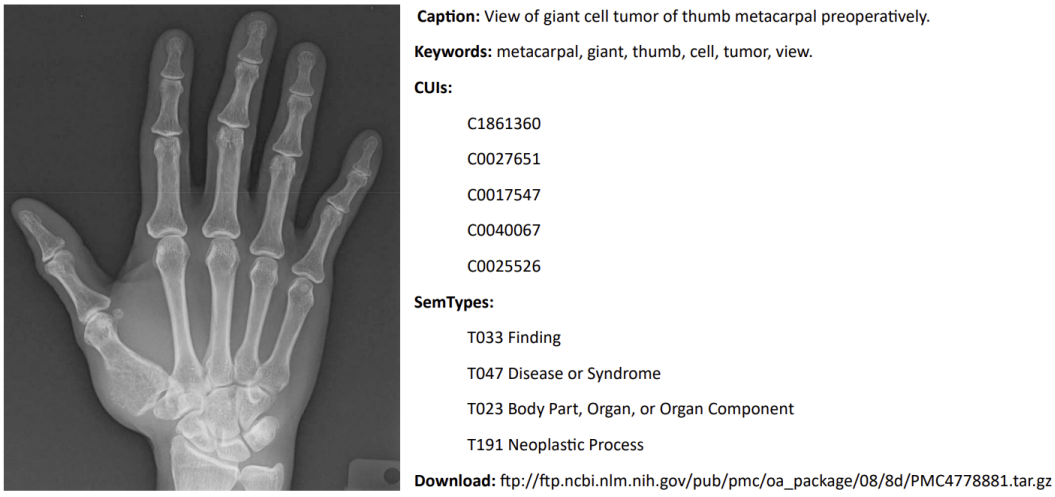
Figura 3 – Imagens pertencem ao subconjunto ‘Não-Radiologia’ do conjunto de dados ROCO



Fonte: Elaboração própria.

Na Figura 4, são ilustradas essas informações de uma imagem qualquer do conjunto.

Figura 4 – Exemplo de uma imagem do conjuto de dados ROCO e suas respectivas informações



Fonte: Elaboração própria.

Utilizando-se o ROCO, é possível criar sistemas para geração de descrições de imagens, o que permite a representação multimodal para conjuntos de imagens que não possuem uma representação textual, sendo este o objetivo principal deste trabalho.

Além disso, é possível criar sistemas com o objetivo de estruturação de imagens e marcação semântica de informações para fins de recuperação de imagens e informações.

3.2 Algoritmo de Seleção de Imagens

Conforme apresentado na seção 3.1, a base de dados ROCO possui aproximadamente 82 mil imagens médicas de diferentes contextos. Portanto, para alcançar o objetivo de criar frases de linguagem natural que descrevam o conteúdo de imagens de médicas, por meio de modelos generativos, o primeiro passo foi criar um algoritmo capaz de selecionar um subconjunto de pares de imagens e descrições textuais de um contexto bem definido da base de dados ROCO.

O algoritmo inicia com a definição de uma função que recebe como entrada um dicionário com nomes de arquivos como chaves e metadados associados como valores e uma lista de palavras-chave para filtrar. Essa função retorna um conjunto de arquivos que correspondem aos critérios do filtro. Um pseudo-código que representa essas ações pode ser visto no Algoritmo 1.

Algoritmo 1: Função get_files()

Input: dictionary: um dicionário de arquivos e suas palavras-chave associadas,
values_of_interest: uma lista de palavras-chave para pesquisar,
operation: uma string indicando se deve retornar arquivos que correspondam a todas as palavras-chave (‘AND’) ou a qualquer palavra-chave (‘OR’).

Output: returned_files: um conjunto de arquivos que correspondem aos critérios de pesquisa

```
1  GetFiles (dictionary, values_of_interest, operation)
2      if operation is not 'AND' and operation is not 'OR' then
3          return 'BAD OPERATION!'
4      end
5      returned_files ← Set()
6      for each key and value in dictionary do
7          add_file ← True
8          for each value_of_interest in values_of_interest do
9              if value_of_interest is in values['keywords'] and add_file then
10                 if operation is 'OR' then
11                     returned_files.add(key)
12                 end
13             end
14             else
15                 if operation is 'AND' then
16                     add_file ← False
17                     break
18                 end
19             end
20         end
21         if operation is 'AND' and add_file then
22             returned_files.add(key)
23         end
24     end
25     return returned_files
26 end
```

Em seguida, é definida uma classe chamada JPEG com um método *decode* que é usado para ler e decodificar arquivos JPEG a fim de eliminar quaisquer imagens corrompidas. Um pseudo-código que representa essas ações pode ser visto no Algoritmo 2.

Algoritmo 2: Algoritmo de decodificação JPEG

Input: Um arquivo de imagem no formato JPEG

Output: Dados JPEG decodificados

```
1  JPEG (image_file)
2      with open(image_file, 'rb') as f:
3      self.img_data ← f.read()
4      def decode()
5          data ← self.img_data
6          while True do
7              marker ← unpack('>H', data[0:2])
8              if marker == 0xffd8 then
9                  data ← data[2 :]
10             end
11             else if marker == 0xffd9 then
12                 return None
13             end
14             else if marker == 0xffda then
15                 data ← data[−2 :]
16             end
17             else
18                 lenchunk ← unpack('>H', data[2:4])
19                 data ← data[2 + lenchunk :]
20             end
21             if len(data) == 0 then
22                 return None
23             end
24         end
25     end
26 end
```

O código então inicializa algumas variáveis e define as localizações dos arquivos

de palavras-chave e descrições para todos os arquivos do conjuntos de dados ROCO. Um pseudo-código que representa essas ações pode ser visto no Algoritmo 3.

Algoritmo 3: Carregamento dos arquivos de palavras-chave e descrições das imagens

Input: Caminho dos arquivos de palavras-chave e descrições das imagens

Output: Palavras-chave e descrições carregadas

```
1  CarregaArquivos ()
2      keywords_file_path_test ← <Caminho da pasta>
3      keywords_file_path_train ← <Caminho da pasta>
4      keywords_file_path_validation ← <Caminho da pasta>
5      captions_file_path_test ← <Caminho da pasta>
6      captions_file_path_train ← <Caminho da pasta>
7      captions_file_path_validation ← <Caminho da pasta>
8      keywords_files_paths ← [keywords_file_path_test, keywords_file_path_train,
         keywords_file_path_validation]
9      captions_files_paths ← [captions_file_path_test, captions_file_path_train,
         captions_file_path_validation]
10     files_paths ← zip(keywords_files_paths, captions_files_paths)
11     return files_paths
12 end
```

Em seguida, o algoritmo mapeia esses arquivos por meio de um dicionário que faz a correspondência entre as imagens e sua localização, palavras-chave e descrições textuais. O código então calcula e imprime o número de palavras únicas no conjunto de dados e suas frequências em ordem decrescente. Um pseudo-código que representa essas ações pode ser visto no Algoritmo 4.

Algoritmo 4: Função que mapeia palavras-chave e descrições aos arquivos

Input: Uma lista de pares de caminhos de arquivo, onde o primeiro caminho é para um arquivo de palavras-chave e o segundo caminho é para um arquivo de descrições.

Output: Mapeia palavras-chave com suas descrições correspondentes e cria um dicionário com elas, onde a chave é o nome da imagem. Também conta as palavras chaves e as exibe em ordem decrescente.

```
1  MapKeywordsCaption ()
2      words ← Counter()
3      dictionary ← {}
4      flat_list ← []
5      for each keywords_file_path and captions_file_path in files_paths do
6          list_words ← []
7          with open(keywords_file_path) as keywords_file,
8                  open(captions_file_path) as captions_file
9          for keywords_line, captions_line in zip(keywords_file, captions_file) do
10             keywords_line ← keywords_line.replace('\t', ' ')
11             key ← keywords_line.split()[0]
12             path ← captions_line.split()[0]
13             keywords ← keywords_line.split()[1:]
14             captions ← captions_line.split()[1:]
15             dictionary[key] ← {'path': path, 'keywords': keywords, 'captions': captions}
16             list_words.append(keywords)
17         end
18         flat_list ← list(chain(*list_words))
19         words.update(flat_list)
20         end
21     end
22     print('Num of words: ' + str(len(words.most_common())))
23     print(words.most_common())
24     return dictionary
25 end
```

No Algoritmo 4, é calculado e impresso, em ordem decrescente, a frequência de todas as palavras-chave de todo o conjunto de dados ROCO. Uma parte dessa impressão pode ser vista na Figura 5.

Figura 5 – Listagem de parte da frequência de todas as palavras-chave do ROCO

```
Num of words: 37075
[('arrow', 14870), ('right', 13396), ('ct', 12752), ('image', 12622), ('left', 10082), ('scan', 9651),
('tomography', 8043), ('radiograph', 7198), ('mas', 7063), ('view', 6911), ('chest', 6881), ('axial', 6819),
('lesion', 6465), ('patient', 5777), ('xray', 5357), ('mri', 4943), ('artery', 4718), ('contrast', 4140),
('coronal', 3868), ('white', 3788), ('lateral', 3612), ('resonance', 3318), ('magnetic', 3310), ('show',
3306), ('abdominal', 3274), ('large', 3259), ('anterior', 3238), ('sagittal', 3025), ('wall', 2998), ('area',
2956), ('abdoman', 2856), ('bilateral', 2835), ('lobe', 2780), ('lung', 2728), ('posterior', 2722), ('bone',
2654), ('small', 2590), ('level', 2525), ('postoperative', 2512), ('upper', 2477), ('tumor', 2387),
('lower', 2374), ('region', 2350), ('multiple', 2286), ('tissue', 2277), ('month', 2235), ('ultrasound',
2198), ('case', 2191), ('fracture', 2162), ('pulmonary', 2100), ('head', 2092), ('normal', 2083), ('brain',
```

Fonte: Elaboração própria.

Como existem 37.075 palavras-chaves distintas com frequências entre 1 e 14.870, definiu-se arbitrariamente um limite inferior para a frequência das palavras, selecionando-se apenas aquelas com frequência maior do que 500. Dessa forma, tem-se apenas as palavras mais relevantes do conjunto ROCO.

Analisando a lista de palavras-chaves remanescentes, verificou-se que era possível agrupar a maioria delas em 3 categorias distintas. As palavras que não foram possíveis de se classificar em alguma dessas categorias foram descartadas. As categorias são:

a) Tipo do Exame;

b) Parte do Corpo;

c) Problema Identificado.

A essas palavras pertencentes as 3 categorias definidas, deu-se a designação de palavras de interesse. Na Tabela 2, são demonstradas as palavras de interesse de cada uma das categorias definidas e suas frequências entre parênteses.

O próximo passo é delimitar o contexto de um subconjunto de imagens do ROCO utilizando as palavras de interesse definidas anteriormente. Primeiramente, as palavras de interesse de cada categoria são inseridas em três listas distintas. Em seguida, o algoritmo cria uma lista de todas as combinações possíveis, tomadas de 3 em 3, dos valores de interesse definidos em cada categoria. Um pseudo-código que representa essas ações pode ser visto no Algoritmo 5.

Tabela 2 – Palavras de Interesse de Cada Categoria

Tipo do Exame	Parte do Corpo	Problema Identificado
‘ct’ (12752)	‘chest’ (6881)	‘tumor’ (2387)
‘scan’ (9651)	‘artery’ (4718)	‘fracture’ (2162)
‘tomography’ (8043)	‘abdoman’ (2856)	‘normal’ (2083)
‘radiograph’ (7198)	‘lobe’ (2780)	‘cystic’ (1700)
‘xray’ (5357)	‘lung’ (2728)	‘cyst’ (1477)
‘mri’ (4943)	‘bone’ (2654)	‘effusion’ (1371)
‘contrast’ (4140)	‘tissue’ (2277)	‘calcification’ (1179)
‘resonance’ (3318)	‘month’ (2235)	‘nodule’ (1158)
‘magnetic’ (3310)	‘pulmonary’ (2100)	‘hepatic’ (997)
‘abdominal’ (3274)	‘head’ (2092)	‘node’ (981)
‘ultrasound’ (2198)	‘brain’ (2038)	‘stent’ (901)
‘angiography’ (1680)	‘vein’ (1897)	‘heterogeneou’ (889)
‘angiogram’ (1220)	‘liver’ (1695)	‘pancreatic’ (874)
‘catheter’ (1020)	‘ventricle’ (1675)	‘week’ (858)
	‘pelvi’ (1635)	‘aneurysm’ (842)
	‘kidney’ (1468)	‘edema’ (833)
	‘spine’ (1321)	‘irregular’ (752)
	‘neck’ (1307)	‘dilatation’ (746)
	‘pleural’ (1294)	‘absces’ (745)
	‘muscle’ (1293)	‘disease’ (743)
	‘renal’ (1266)	‘apical’ (700)
	‘coronary’ (1258)	‘hematoma’ (688)
	‘femoral’ (1252)	‘fistula’ (608)
	‘cervical’ (1196)	‘cancer’ (505)
	‘atrium’ (1192)	
	‘bowel’ (1171)	
	‘aorta’ (1098)	
	‘aortic’ (1064)	
	‘hip’ (1018)	
	‘heart’ (620)	
	‘tooth’ (619)	

Fonte: Elaboração própria.

Algoritmo 5: Definição da lista de combinações das palavras de interesse

Input: Palavras de interesse de cada categoria

Output: Lista com todas as combinações tomadas de 3 em 3 das palavras de interesse

```
1  DefinicaoCombinacoesPalavrasInteresse ()
2      types_exam ← ['ct', 'scan', 'tomography', 'radiograph', 'xray', 'mri', 'contrast',
        'resonance', 'magnetic', 'abdominal', 'ultrasound', 'angiography', 'angiogram',
        'catheter']
3      bodys_parts ← ['chest', 'artery', 'abdoman', 'lobe', 'lung', 'bone', 'tissue', 'month',
        'pulmonary', 'head', 'brain', 'vein', 'liver', 'ventricle', 'pelvi', 'kidney', 'spine',
        'neck', 'pleural', 'muscle', 'renal', 'coronary', 'femoral', 'cervical', 'atrium',
        'bowel', 'aorta', 'aortic', 'hip', 'heart', 'tooth']
4      problems ← ['tumor', 'fracture', 'normal', 'cystic', 'cyst', 'effusion', 'calcification',
        'nodule', 'hepatic', 'node', 'stent', 'heterogeneou', 'pancreatic', 'week',
        'aneurysm', 'edema', 'irregular', 'dilatation', 'absces', 'disease', 'apical',
        'hematoma', 'fistula', 'cancer']
5      all_values_of_interest ← []
6      for type in types_exam do
7          for body in bodys_part do
8              for problem in problems do
9                  all_values_of_interest.append([type, body, problem])
10             end
11         end
12     end
13     return (all_values_of_interest)
14 end
```

Para cada combinação de palavras de interesse, o algoritmo procura imagens que correspondam a esses valores no dicionário criado pelo Algoritmo 4 e as salva em um diretório especificado.

Mais especificamente, o algoritmo itera sobre a lista de todas as combinações possíveis e procura imagens que correspondam à combinação atual de valores de interesse. Se o número de imagens correspondentes encontradas na base de dados ROCO estiver entre

10 e 80, o algoritmo cria um novo diretório o qual ele copia as imagens como arquivos JPEG, redimensionando-as para uma largura de 500 *pixels*.

O algoritmo então gera três arquivos de texto para o conjunto de imagens selecionadas com base no contexto: *1_keywords.txt*, *2_captions.txt* e *3_valofint.txt*. O arquivo *1_keywords.txt* contém as palavras-chave associadas as imagens, o arquivo *2_captions.txt* contém as descrições associadas as imagens e o arquivo *3_valofint.txt* contém a combinação de valores que levaram à seleção de cada imagem. Também é gerado um arquivo *0_values_of_interest.txt* que contem a contagem de arquivos de cada uma das combinações de palavras de interesse possíveis e a soma total de arquivos de todas as combinações buscadas.

É importante destacar que caso uma imagem pertença a diferentes combinações de palavras de interesse, ela será atribuída a primeira combinação que ocorrer na iteração. Por exemplo, se uma imagem qualquer for encontrada nas combinações [*'ct', 'chest', 'tumor'*] e [*'scan', 'chest', 'tumor'*], processadas em sequência na iteração de todas as combinações, esta imagem será considerada como pertencente a primeira combinação iterada, ou seja, [*'ct', 'chest', 'tumor'*].

Um pseudocódigo com toda essa lógica de seleção de imagens do contexto desejado pode ser visualizado no Algoritmo 6.

Algoritmo 6: Separa arquivos de imagem e texto dado o contexto de interesse

Input: Objetos *dictionary* e *all_values_of_interest* criados anteriormente.

Output: Arquivos de imagem e texto separados no diretório de destino.

```
destination = <Caminho da pasta de saída>
total_files = 0
for values_of_interest in all_values_of_interest do
    files = sorted(get_files(dictionary, values_of_interest, 'AND'))
    if 10 <= len(files) <= 80 then
        total_files = total_files + len(files)
        with open(file='0_values_of_interest.txt') as values_of_interest_file:
        values_of_interest_file.write(values_of_interest + ' ' + str(len(files)) + '\n')
        for file in files do
            source = dictionary.get(file)['path']
            try
                image = JPEG(source).decode()
                if (image not in destination) then
                    copy2(source, destination)
                    img = Image.open(destination + file + '.jpg')
                    img = img.resize('500px')
                    img.save(destination + file + '.jpg')
                    with open(file='1_keywords.txt') as keywords_file,
                            open(file='2_captions.txt') as captions_file,
                            open(file='3_valofint.txt') as valofint_file:
                    keywords_file.write(file + ' ' + dictionary.get(file)['keywords'] + '\n')
                    captions_file.write(file + ' ' + dictionary.get(file)['captions'] + '\n')
                    valofint_file.write(file + ' ' + values_of_interest + '\n')
                end
            except
                print('Não foi possível copiar o arquivo: ' + source)
            end
        end
    end
end
with open(file='0_values_of_interest.txt') as values_of_interest_file:
values_of_interest_file.write('Total = ' + str(total_files) + '\n')
```

3.3 Descrição do Experimento

Nesta seção, é descrito em detalhes o experimento realizado com o intuito de se criar um modelo generativo de textos descritivos de imagens médicas. O objetivo principal do experimento foi investigar as combinações possíveis das variáveis estudadas, a fim de obter *insights* valiosos sobre o desempenho do modelo.

Para extrair os atributos das imagens, utilizou-se uma variedade de modelos de reconhecimento de imagens clássicas: MobileNetV2, DenseNet201, ResNet152V2, NASNetLarge, VGG19, Xception, InceptionV3 e InceptionResNetV2. Cada modelo foi empregado individualmente para avaliar sua capacidade de capturar informações relevantes nas imagens médicas.

Além disso, explorou-se diferentes otimizadores de gradiente para o treinamento dos modelos: Adam, AdamW, Adadelta e Adafactor. Com isso, buscou-se determinar qual otimizador proporcionaria os melhores resultados em termos de desempenho computacional e convergência do modelo.

Outra linha de experimentação envolveu o uso do conceito de *Transfer Learning* nos extratores de atributos das imagens. Portanto, em parte das experimentações, utilizou-se extratores de atributos pré-treinados com o conjunto de dados *ImageNet*. Além disso, em se tratando de *Transfer Learning*, testou-se casos com e sem o ajuste dos pesos durante a etapa de treinamento, com o intuito de avaliar a influência desse processo nos resultados obtidos. Também realizou-se experimentos sem o uso do *Transfer Learning* do *ImageNet*, buscando explorar a capacidade do modelo em aprender diretamente com os dados médicos disponíveis.

Durante o experimento, também considerou-se a aplicação de técnicas de aumento de dados, conhecidas como *image augmentation*. Essas técnicas têm o propósito de ampliar o conjunto de dados disponíveis, aplicando transformações como rotação, zoom e espelhamento nas imagens. Realizou-se experimentos com e sem o uso de aumento de dados, visando analisar o impacto dessa técnica no desempenho do modelo.

É importante ressaltar que todos os hiper-parâmetros de cada seção de testes foram otimizados por meio da API KerasTuner, utilizando a abordagem de *Grid Search*. Esse processo permitiu uma exploração sistemática das combinações de hiper-parâmetros e auxiliou na identificação das melhores configurações para cada modelo.

Durante o treinamento de cada variante, monitorou-se o desempenho dos modelos através da acurácia, bem como o tempo de processamento envolvido, medido pelo tempo

necessário para o treinamento e teste. O melhor modelo é definido como o de maior acurácia.

Com o intuito de abordar as possíveis limitações da solução proposta, decidiu-se adicionar dados desbalanceados ao conjunto de treinamento, validação e teste. Essa escolha permitiu analisar o impacto do desequilíbrio de classes no desempenho do modelo e investigar o tamanho mínimo do conjunto de dados necessário para alcançar resultados satisfatórios. Essa análise foi fundamental para fornecer *insights* sobre a capacidade do modelo de lidar com dados desbalanceados com eficácia.

Por fim, para o melhor modelo obtido, realizou-se uma avaliação detalhada de seu desempenho por meio da construção de matrizes de confusão de cada parte dos textos gerados (tipo de exame, parte do corpo e problema identificado). Essas matrizes permitiram visualizar as escolhas corretas e incorretas feitas pelo melhor modelo na construção dos textos descritivos das imagens médicas. Para cada uma das matrizes, calculou-se indicadores-chave, como precisão, taxa de acerto, medida F1 e outros. Esses indicadores forneceram uma medida objetiva da qualidade das descrições textuais geradas pelo modelo e serviram para avaliar seu desempenho.

Ao combinar todas as variáveis estudadas e realizar experimentos sistemáticos, pôde-se explorar diversas combinações de modelos de extração de atributos, otimizadores de gradiente, uso do *Transfer Learning* (com e sem ajuste dos pesos), técnicas de aumento de dados e a presença de dados desbalanceados. Com essa abordagem, foi possível obter uma compreensão aprofundada do desempenho, tempo de processamento e limitações da solução proposta.

3.4 Algoritmo Generativo de Textos Descritivos de Imagens Médicas

Esta seção apresenta a metodologia referente ao algoritmo generativo construído utilizando técnicas de *deep learning* e linguagem natural para a geração de descrições de imagens médicas. As principais etapas são:

a) Definição de Parâmetros do Ambiente;
b) Preparação do Conjunto de Dados Textuais;
c) Limpeza e Vetorização dos Dados de Texto;
d) Construção da *Pipeline* para Treinamento;
e) Construção do Modelo;
f) Treinamento do Modelo;
g) Geração de Descrições de Imagens Médicas.

A seguir, serão apresentados os detalhes de cada uma dessas etapas, destacando os

principais pontos de cada uma delas.

3.4.1 Definição de Parâmetros do Ambiente

No Algoritmo 7, são definidos os parâmetros utilizados no decorrer do algoritmo de geração de descrições de imagens médicas.

Ele começa definindo uma semente para garantir que os resultados sejam reproduzíveis. Em seguida, especifica o caminho para o diretório onde as imagens estão localizadas e as dimensões desejadas para essas imagens. O tamanho do vocabulário, ou seja quantas palávras diferentes é possível vetorizar, é definido como 50 e o tamanho máximo permitido de palavras para qualquer sequência textual é 5.

As dimensões das representações internas (*embedding*) de imagem e *token*, bem como as unidades por camada na rede *feed-forward* são definidos em seguida. O modelo também usa 2 cabeças de atenção para o codificador e decodificador. Também é definido uma variável lógica que determina se os pesos iniciais utilizados na rede neural convolucional da extração de características das imagens podem ser ajustadas na etapa de treinamento do modelo.

Outros parâmetros, como o tamanho do lote de dados de treinamento, o número máximo de épocas de treinamento e o número de descrições textuais por imagem são definidos posteriormente.

Algoritmo 7: Definição dos parâmetros do ambiente

Input: Parâmetros informados pelo usuário.

Output: Parâmetros disponíveis para as próximas etapas do algoritmo.

```
1  DefineParametros ()
2      SEED ← 1337
3      IMAGES_PATH ← <Caminho da pasta das imagens>
4      IMAGE_SIZE ← (500, 500)
5      VOCAB_SIZE ← 50
6      SEQ_LENGTH ← 5
7      EMBED_DIM ← 512
8      FF_DIM ← 512
9      ENCODER_NUM_HEADS ← 2
10     DECODER_NUM_HEADS ← 2
11     BASE_MODEL_TRAINABLE ← True
12     BATCH_SIZE ← 64
13     EPOCHS ← 150
14     NUM_CAPTIONS_PER_IMAGE ← 1
15 end
```

3.4.2 Preparação do Conjunto de Dados Textuais

No Algoritmo 8, é preparado o conjunto de descrições das imagens médicas, realizando a organização, formatação e mapeamento desses dados.

Primeiramente, uma função chamada ‘load_captions_data’ recebe um parâmetro ‘filename’, que deve ser o caminho para um arquivo de texto contendo descrições associadas as imagens.

A função carrega os dados do arquivo de texto e cria um dicionário chamado ‘captions_mapping’ que mapeia os nomes das imagens e suas respectivas descrições. Ele também cria uma lista chamada ‘text_data’ que contém todas as descrições disponíveis.

Em seguida, o código itera sobre cada linha do arquivo de texto e separa o nome da imagem e a descrição usando um caractere de tabulação. Ele também remove o sufixo de número de descrição dos nomes de imagens e adiciona o caminho para as imagens que devem estar em uma pasta definida em ‘IMAGES_PATH’.

O código então adiciona um *token* de início e um de fim a cada descrição e o adiciona à lista ‘text_data’. Em seguida, ele verifica se a imagem já está no dicionário ‘captions_mapping’. Se já estiver, ele adiciona a descrição à lista de descrições associadas à imagem. Caso contrário, ele cria uma nova entrada no dicionário.

Por fim, a função retorna o dicionário ‘captions_mapping’ que consiste em um dicionário que mapeia os nomes das imagens com as descrições correspondentes e uma lista ‘text_data’ contendo todas as descrições disponíveis.

Algoritmo 8: Carrega dados de descrição e mapeia para as imagens correspondentes

Input: filename: Caminho para o arquivo de texto contendo as descrições.

Output: captions_mapping:Dicionário que mapeia os nomes das imagens com as descrições correspondentes; text_data: Lista contendo todas as descrições.

```
1  LoadCaptionsData (filename)
2      import os
3      captions_mapping ← {}
4      text_data ← []
5      caption_file ← readlines(filename)
6      for line in caption_file do
7          line ← line.rstrip('\n')
8          img_name, caption ← line.split('\t')
9          img_name ← img_name.split('#')[0]
10         img_name ← os.path.join(IMAGES_PATH, img_name.strip())
11         caption ← '<start> ' + caption.strip() + ' <end>'
12         text_data.append(caption)
13         if img_name in captions_mapping then
14             captions_mapping[img_name].append(caption)
15         end
16         else
17             captions_mapping[img_name] ← [caption]
18         end
19     end
20     return captions_mapping, text_data;
21 end
```

Em seguida, no Algoritmo 9, é recebido o dicionário contendo os dados das descri-

ções das imagens mapeadas que são divididas nos conjuntos de treinamento e teste. É possível ajustar o tamanho do conjunto de treinamento com o argumento 'train_size', que é um valor que representa a fração do conjunto de dados a ser usada para o treinamento. O valor padrão é 0,8, o que significa que 80% do conjunto de dados será usado para o treinamento. Além disso, a opção 'shuffle', de valor padrão *True*, permite que o usuário embaralhe o conjunto de dados antes de fazer a divisão, o que pode ajudar a evitar vieses nos conjuntos de dados.

No Algoritmo 9, primeiramente é obtido uma lista de todos os nomes de imagens no dicionário 'caption_data' que são embaralhados se a opção 'shuffle' for verdadeira.

Em seguida, divide essa lista em conjuntos de treinamento e teste, com o tamanho do conjunto de treinamento definido como uma fração do tamanho total do conjunto de dados.

Por fim, a função retorna dois dicionários contendo os dados de treinamento e teste separados.

Algoritmo 9: Divisão dos dados em conjuntos de treinamento e teste

Input: caption_data: Dicionário contendo os dados de legendas mapeados;
train_size: Fração do conjunto de dados a ser usada como treinamento;
shuffle: Se deve embaralhar o conjunto de dados antes da divisão.

Output: Dicionários contendo os conjuntos de treinamento e teste separados

```
1  TrainTestSplit (caption_data, train_size ← 0.8, shuffle ← True)
2      all_images ← list(caption_data.keys())
3      if shuffle then
4          Shuffle(all_images)
5      end
6      train_size ← int(len(caption_data) * train_size)
7      training_data = {img_name: caption_data[img_name] for img_name in
         all_images[:train_size]}
8      test_data = {img_name: caption_data[img_name] for img_name in
         all_images[train_size:]}
9      return training_data, test_data
10 end
```

3.4.3 Limpeza e Vetorização dos Dados de Texto

O próximo passo é limpar e vetorizar os dados de texto para que o modelo possa trabalhar com eles adequadamente.

No Algoritmo 10, é definida uma função chamada *'custom_standardization'* que realiza a padronização de uma *string* de entrada, transformando todas as letras em minúsculas e removendo qualquer caractere especial presente na variável *'strip_chars'*, utilizando uma Expressão Regular (Regex).

Em seguida, é criado um objeto *'TextVectorization'*, importado do *TensorFlow*, que utiliza a função *'custom_standardization'* para padronizar o texto de entrada. Esse objeto possui os parâmetros *'max_tokens'*, que define a quantidade máxima de diferentes palavras que podem existir no vocabulário do vetorizador, *'output_mode'*, que consiste no modo de saída para converter o texto e *'output_sequence_length'*, que define o comprimento máximo das sequências de saída.

O *'output_mode'* é definido como 'int', o que significa que as textos originais vão ser representadas por sequências de números inteiros, onde cada inteiro representa o índice de uma palavra em um vocabulário. Os parâmetros *'max_tokens'* e *'output_sequence_length'* são definidos pelas constantes VOCAB_SIZE e SEQ_LENGTH determinados no Algoritmo 7.

Por fim, o objeto *'vectorization'*, por meio do método *'adapt'*, recebendo como argumento a lista *'text_data'* gerado pelo Algoritmo 8, gera as representações vetoriais das palavras, considerando os parâmetros passados em sua inicialização.

Algoritmo 10: Limpeza e Vetorização dos Dados de Texto

Input: Dados textuais brutos.

Output: Textos limpos e vetorizados.

```
LimpezaVetorizacaoTextos (text_data)
    from tensorflow.keras.layers import TextVectorization
    CustomStandardization (input_string)
        lowercase ← strings.lower(input_string)
        strip_chars ← '!\'#$%&'()*+,-./:;<=>?@[]^_`{|}~'
        strip_chars ← strip_chars.replace('<', '')
        strip_chars ← strip_chars.replace('>', '')
        return regex_replace(lowercase, '[%s]' % regex.escape(strip_chars), '')
    end
    vectorization ← TextVectorization(
                        max_tokens ← VOCAB_SIZE,
                        output_mode ← 'int',
                        output_sequence_length ← SEQ_LENGTH,
                        standardize ← custom_standardization,
    )
    vectorization.adapt(text_data)
    return vectorization
end
```

3.4.4 Construção da Pipeline para Treinamento

Em seguida, foi construída uma *pipeline* para o treinamento do modelo. No Algoritmo 11, é definido o fluxo do processamento dos dados dessa *pipeline*, no qual é aplicado transformações nas imagens e textos, são construídos os lotes dos conjuntos de dados de treinamento e teste e definido o paralelismo do processamento.

A *pipeline* se divide em três funções que são usadas para preparar e criar conjuntos de dados *TensorFlow* a serem utilizados na criação de um modelo generativo de descrições textuais de imagens.

A primeira função, *decode_and_resize*, recebe um caminho de um arquivo de imagem como entrada, lê o arquivo com a biblioteca *TensorFlow*, decodifica a imagem como um

arquivo JPEG com 3 canais de cor, redimensiona a imagem para um tamanho especificado em IMAGE_SIZE, definido no Algoritmo 7, e converte a imagem para o formato *float32*. Dessa forma, tem-se ao final dessa primeira etapa da *pipeline* uma imagem pré-processada.

A segunda função, *process_input*, combina e retorna as imagens pré-processadas e suas respectivas descrições vetorizadas pelo Algoritmo 10. Ela recebe como entrada o caminho de um arquivo de imagem e a sua respectiva descrição associadas.

A terceira função, *make_dataset*, usa a função *from_tensor_slices* do *TensorFlow* para criar conjuntos de dados das imagens e descrições transformadas anteriormente. Ela embaralha o conjunto de dados, agrupa os elementos em lotes com um tamanho definido na constante BATCH_SIZE, definido no Algoritmo 7, e organiza o paralelismo dos lotes para melhorar o desempenho do treinamento do modelo. Dessa forma, tem-se conjuntos de dados pré-processados e preparados para uso.

Algoritmo 11: Construção da Pipeline Para Treinamento

Input: Imagens e textos brutos.

Output: Conjuntos de dados *TensorFlow* pré-processados e preparados para uso.

```
PipeLineTreinamento ()
    import TensorFlow as tf
    decode_and_resize (img_path)
        img ← tf.io.read_file(img_path)
        img ← tf.image.decode_jpeg(img, channels ← 3)
        img ← tf.image.resize(img, IMAGE_SIZE)
        img ← tf.image.convert_image_dtype(img, tf.float32)
        return img
    end
    process_input (img_path, captions)
        return decode_and_resize(img_path), vectorization(captions)
    end
    make_dataset (images, captions)
        dataset ← tf.data.Dataset.from_tensor_slices((images, captions))
        dataset ← dataset.shuffle(BATCH_SIZE * 8)
        dataset ← dataset.map(process_input, num_parallel_calls ← AUTOTUNE)
        dataset ← dataset.batch(BATCH_SIZE).prefetch(AUTOTUNE)
        return dataset
    end
    train_dataset ← make_dataset(list(train_data.keys()), list(train_data.values()))
    test_dataset ← make_dataset(list(test_data.keys()), list(test_data.values()))
    return train_dataset, test_dataset
end
```

3.4.5 Construção do Modelo

Com a *pipeline* pronta, foi construído um modelo generativo de descrições de imagens médicas utilizando uma CNN para a extração de características das imagens, um codificador Transformador para gerar um vetor que represente essas características de uma forma mais relevante ao modelo e um decodificador Transformador para o NLP. O modelo é composto

por quatro partes bem definidas, que serão detalhadas nas subseções a seguir:

a) Construção da Rede Neural Convolucional;

b) Construção do Codificador Transformador;

c) Construção do Decodificador Transformador;

d) Construção do Modelo Generativo de Descrições de Imagens.

3.4.5.1 Construção da Rede Neural Convolucional

Na primeira parte da construção do modelo, é definido uma CNN usando uma das arquitetura estudadas, importada do *TensorFlow*, como modelo base, inicializado com os pesos pré-treinados no conjunto de dados *ImageNet*, se o *Transfer Learning* for utilizado. Essa CNN é usada para extrair as características das imagens de entrada.

No Algoritmo 12, é definida uma função que retorna uma instância de uma CNN, que é posteriormente usada no modelo generativo. Vale destacar que a variável *trainable* determina se os pesos das camadas do modelo base serão ajustáveis no processo de treinamento ou se permanecerão os definidos pelo *ImageNet*, no caso de utilização do *Transfer Learning*. Essa condição é tratada pela constante *BASE_MODEL_TRAINABLE* definida no Algoritmo 7.

Algoritmo 12: Construção da Rede Neural Convolucional

Input: Instancia da CNN utilizada.

Output: CNN configurada para extrair as características das imagens.

```
1  get_cnn_model ()
2      from tensorflow.keras.applications import efficientnet
3      from tensorflow import keras, keras.layers, keras.models
4      base_model ← <Instancia da CNN analisada>
5      base_model.trainable ← BASE_MODEL_TRAINABLE
6      base_model_out ← base_model.output
7      base_model_out ← layers.Reshape((-1,
         base_model_out.shape[-1]))(base_model_out)
8      cnn_model ← keras.models.Model(base_model.input, base_model_out)
9      return cnn_model
10 end
```

3.4.5.2 Construção do Codificador Transformador

Na segunda parte da construção do modelo, é definido um codificador Transformador que é utilizado para codificar os atributos das imagens e gerar uma representação vetorial dessas informações.

Para a construção desse codificador, definiu-se uma classe chamada *Transformer Encoder Block*. Esse codificador possui uma camadas de atenção, camadas de normalizações e uma camada densa que, em conjunto, transformam a entrada em uma representação mais útil para o modelo. Dessa forma, esse codificador possui dois hiper-parâmetros que estão definidos no Algoritmo 7:

a) *embed_dim*: a dimensão dos vetores que representam as imagens;

b) *num_heads*: o número de cabeças a serem usadas na camada de atenção.

No Algoritmo 13, é utilizada a *Framework TensorFlow* e implementado o codificador Transformador. Ele tem como entrada atributos das imagens pré-processadas anteriormente que são passadas ao parâmetro *input* e uma parâmetro lógico *training* que indica se o codificador está sendo usado em uma etapa de treinamento ou não.

Primeiramente, a entrada é normalizada usando a camada *Layer Normalization* e, em seguida, passada pela camada *Dense*. Essa camada consiste em uma camada densa totalmente conectada com uma função de ativação do tipo Unidade Linear Retificada, do inglês 'Rectified Linear Unit' (ReLU). O resultado é então passado pela camada de atenção *Multi Head Attention* com várias cabeças que retorna um tensor *attention_output*. Esse tensor é adicionada à entrada original e normalizada pela camada *Layer Normalization*. O resultado final é uma representação vetorial dos atributos das imagens com valor ao modelo generativo. Esse vetor, por fim, é retornado como a saída do codificador Transformador.

Algoritmo 13: Construção do Codificador Transformador

Input: Atributos das imagens pré-processadas anteriormente.

Output: Representação vetorial das informações dos atributos das imagens com valor ao modelo.

```
1 Class TransformerEncoderBlock (inputs, training)
2     from tensorflow.keras import layers
3     inputs ← layers.LayerNormalization()(inputs)
4     inputs ← layers.Dense(dense_dim ← EMBED_DIM, activation ← 'relu')(inputs)
5     attention_output ← layers.MultiHeadAttention(num_heads ←
        ENCODER_NUM_HEADS, key_dim ← EMBED_DIM, dropout ← 0.0)(query ←
        inputs, value ← inputs, key ← inputs, training ← training)
6     outputs ← layers.LayerNormalization()(inputs + attention_output)
7     return outputs
8 end
```

3.4.5.3 Construção do Decodificador Transformador

Na terceira parte da construção do modelo, é definido um decodificador Transformador. Esse bloco é composto por várias camadas, incluindo camadas *Multi Head Attention*, camadas de normalização e camadas densas. O decodificador recebe como entrada um tensor com os *tokens* dos textos e as saídas produzidas pelo codificador e realiza uma série de operações a elas, retornando como saída uma distribuição de probabilidade sobre o vocabulário. Esse decodificador possui 4 hiper-parâmetros que estão definidos no Algoritmo 7:

a) *embed_dim*: a dimensão dos vetores que representam cada palavra do vocabulário;
b) *ff_dim*: unidades por camada na rede *feed-forward*;
c) *num_heads*: o número de cabeças a serem usadas na camada de atenção;
d) *vocab_size*: o tamanho do vocabulário, ou seja quantas palávras diferentes é possível vetorizar.

Para a construção desse decodificador, definiu-se uma classe chamada *Transformer Decoder Block*. Essa classe possui três métodos:

a) *get_positional_embedding()*;
b) *get_causal_attention_mask()*;

c) *call()*.

No Algoritmo 14 é implementado o método *get_positional_embedding()* que consiste em uma função que calcula a representação interna posicional para os *tokens* de entrada. Esse método recebe como entrada um tensor *inputs* que representa os *tokens* de entrada. Ele começa calculando o comprimento da sequência de entrada usando o método *tf.shape()* e, em seguida, cria um vetor de posição usando o método *tf.range()*, ambos do *TensorFlow*. Depois disso, é utilizada a classe *Embedding* do *TensorFlow* para criar duas representações internas, uma para os *tokens* de entrada e outra para os vetores de posição. A representação interna dos *tokens* é multiplicada pela raiz quadrada da dimensão da representação interna e, então, as duas representações são somadas e retornadas como a representação final dos *tokens* de entrada.

Algoritmo 14: Decodificador Transformador - Método *get_positional_embedding()*

Input: Tensor *inputs* que representa os *tokens* de entrada.

Output: Representação interna posicional (*embedding*) dos *tokens* de entrada.

```
1  Class TransformerDecoderBlock ()
2      import tensorflow as tf
3      from tensorflow.keras import layers
4      get_positional_embedding (inputs)
5          length ← tf.shape(inputs)[-1]
6          positions ← tf.range(start ← 0, limit ← length, delta ← 1)
7          embedded_tokens ← layers.Embedding(input_dim ← VOCAB_SIZE,
             output_dim ← EMBED_DIM)(inputs)
8          embedded_tokens ← embedded_tokens * tf.math.sqrt(tf.cast(EMBED_DIM,
             tf.float32))
9          embedded_positions ← layers.Embedding(input_dim ← SEQ_LENGTH,
             output_dim ← EMBED_DIM)(positions)
10         positional_embedding ← embedded_tokens + embedded_positions
11         return positional_embedding
12     end
13 end
```

No Algoritmo 15 é implementado o método *get_causal_attention_mask()* que consiste em uma função que constrói uma máscara de atenção causal, que é uma matriz booleana triangular inferior usada na camada *Multi Head Attention* para limitar a atenção apenas aos *tokens* passados e ao *token* atual, garantindo que o modelo não possa prever *tokens* futuros. Esse

método recebe como entrada um tensor *inputs* que representa os *tokens* de entrada. Para alcançar esse objetivo, utiliza-se o método *tf.range()* do *TensorFlow* para criar um vetor com as posições na sequência de entrada. Em seguida, cria-se uma matriz booleana onde cada valor é *True* se a posição atual for maior ou igual à posição correspondente na sequência e *False* caso contrário. A máscara é então convertida para o tipo *int32* e retornada como um tensor.

Algoritmo 15: Decodificador Transformador - Método *get_causal_attention_mask()*

Input: Tensor *inputs* que representa os *tokens* de entrada.

Output: Tensor que representa uma máscara de atenção causa

```
1  Class TransformerDecoderBlock ()
2      import tensorflow as tf
3      get_causal_attention_mask (inputs)
4          input_shape ← tf.shape(inputs)
5          batch_size, sequence_length ← input_shape[0], input_shape[1]
6          i ← tf.range(sequence_length)[:, tf.newaxis]
7          j ← tf.range(sequence_length)
8          mask ← tf.cast(i >= j, dtype='int32')
9          mask ← tf.reshape(mask, (1, input_shape[1], input_shape[1]))
10         mult ← tf.concat([tf.expand_dims(batch_size, -1), tf.constant([1, 1], dtype ←
            tf.int32)], axis ← 0)
11         return tf.tile(mask, mult)
12     end
13 end
```

No Algoritmo 16 é implementado o método *call()* que consiste no método principal que é chamado quando um decodificador do Transformador é instanciado. Ele tem como parâmetros:

a) *inputs*: tensor com os *tokens* de entrada;
b) *encoder_outputs*: a saída do *encoder* dado uma imagem;
c) *mask*: um tensor de máscara;
d) *training*: parâmetro lógico que indica se o decodificador está sendo usado em uma etapa de treinamento ou não.

O método começa chamando o *get_positional_embedding()* para calcular a representação interna posicional dos *tokens* de entrada. Em seguida, *get_causal_attention_mask()* é chamado para se obter a máscara de atenção causal. Então, é criada uma máscara de *padding* e

uma máscara combinada usando o tensor que foi passado como argumento ao parâmetro *mask* quando o decodificador foi instanciado, que são convertidas para o tipo *int32*. Em seguida, a função *tf.minimum()* do *TensorFlow* é utilizada entre a máscara combinada e a máscara causal para se obter a máscara de atenção que é utilizada na primeira camada de *Multi Head Attention* do *TensorFlow* encadeada logo a seguir.

A primeira camada de *Multi Head Attention* recebe a representação interna posicional dos *tokens* de entrada e a máscara de atenção combinada e possui um *dropout* de 0,1. A saída dessa camada é somada à representação interna posicional dos *tokens* de entrada e normalizada usando a camada *Layer Normalization* do *TensorFlow*.

A segunda camada *Multi Head Attention* recebe a saída da camada *Layer Normalization* anterior, a saída do codificador Transformador e a máscara de atenção *padding* e possui um *dropout* de 0,1. Então, é gerada uma saída que é adicionada à saída da primeira camada *Layer Normalization* e normalizada novamente em uma segunda camada *Layer Normalization*.

Em seguida, é construída uma camada *feed forward* através do encadeamento de uma camada *Dense* com uma função de ativação ReLU, uma camada *Dropout*, parametrizada com 0,3, e outra camada *Dense*, todas do *TensorFlow*. Esse encadeamento recebe como entrada a saída da *Layer Normalization* anterior.

A seguir, uma terceira camada *Layer Normalization* recebe a saída da camada *feed forward* somada a saída da *Layer Normalization* anterior e é encadeada a uma outra camada *Dropout*, parametrizada com 0,5, e outra camada *Dense* usando a função de ativação *softmax* que gera a saída final como uma distribuição de probabilidade.

A saída final é então retornada e consiste em uma matriz com a mesma forma que a entrada (*inputs*), onde cada elemento é uma probabilidade associada a um *token* do vocabulário.

Algoritmo 16: Decodificador Transformador - Método *call()*

Input: *inputs*: tensor com os *tokens* de entrada; *encoder_outputs*: a saída do codificador Transformador dado uma imagem de entrada; *mask*: um tensor de máscara passado na instanciação do decodificador; *training*: parâmetro lógico que indica se o decodificador está sendo usado em uma etapa de treinamento ou não.

Output: Distribuição de probabilidade sobre o vocabulário.

```
Class TransformerDecoderBlock ()
    import tensorflow as tf
    from tensorflow.keras import layers
    call (inputs, encoder_outputs, training, mask)
        positional_embedding ← self.get_positional_embedding(inputs)
        causal_mask ← self.get_causal_attention_mask(positional_embedding)
        padding_mask ← tf.cast(mask[:, :, tf.newaxis], dtype ← tf.int32)
        combined_mask ← tf.cast(mask[:, tf.newaxis, :], dtype ← tf.int32)
        combined_mask ← tf.minimum(combined_mask, causal_mask)
        attention_output_1 ← layers.MultiHeadAttention(num_heads ← DECODER_NUM_HEADS,
          key_dim ← EMBED_DIM, dropout ← 0.1)( query ← positional_embedding, value ←
          positional_embedding, key ← positional_embedding, attention_mask ← combined_mask,
          training ← training)
        normalization_1 ← layers.LayerNormalization()(positional_embedding + attention_output_1)
        attention_output_2 ← layers.MultiHeadAttention(num_heads ← DECODER_NUM_HEADS,
          key_dim ← EMBED_DIM, dropout ← 0.1)( query ← normalization_1, value ←
          encoder_outputs, key ← encoder_outputs, attention_mask ← padding_mask, training ←
          training)
        normalization_2 ← layers.LayerNormalization()(normalization_1 + attention_output_2)
        ffn_out ← layers.Dense(FF_DIM, activation ← 'relu')(normalization_2)
        ffn_out ← layers.Dropout(0.3)(ffn_out, training ← training)
        ffn_out ← layers.Dense(EMBED_DIM)(ffn_out)
        ffn_out ← layers.LayerNormalization()(ffn_out + normalization_2, training ← training)
        ffn_out ← layers.Dropout(0.5)(ffn_out, training ← training)
        outputs ← layers.Dense(VOCAB_SIZE, activation ← 'softmax')(ffn_out)
        return outputs
    end
end
```

3.4.5.4 Construção do Modelo Generativo de Descrições de Imagens

Por fim, é definida a classe *Image Captioning Model* que combina todas as outras 3 partes descritas anteriormente para construir o modelo generativo propriamente dito. Em resumo,

essa classe implementa um modelo de geração de descrições de imagens médicas que usa uma rede neural convolucional para extrair os atributos das imagens, um codificador Transformador para representar vetorialmente os atributos das imagens fornecidas e um decodificador Transformador para gerar a distribuição de probabilidade sobre o vocabulário. Vale ressaltar que essa classe herda da classe *Model* do *Tensorflow*.

O construtor dessa classe recebe quatro parâmetros:

a) *cnn_model*: modelo de rede neural convolucional que é utilizado para extrair os atributos das imagens;
b) *encoder*: codificador Transformador para representar vetorialmente os atributos das imagens;
c) *decoder*: decodificador Transformador que retornando uma distribuição de probabilidade sobre o vocabulário;
d) *num_captions_per_image*: número de descrições textuais fornecidas para cada imagem de entrada;
e) *image_aug*: função de aumento de dados (*data augmentation*) para imagens.

No Algoritmo 17, é implementado o construtor da classe *Image Captioning Model* que, além de inicializar o modelo generativo com as referências passadas, define duas métricas de avaliação de desempenho. A primeira delas é a perda, que representa a medida do erro entre as previsões do modelo e os valores reais do conjunto de dados. A segunda métrica é a acurácia, definida como a medida da precisão do modelo em acertar as previsões em relação aos valores reais. Essas métricas são registradas, respectivamente, nos objetos *loss_tracker* e *acc_tracker*, importados do *TensorFlow*.

Algoritmo 17: Classe *Image Captioning Model* - Construtor

Input: *CNN*, Codificador Transformador, Decodificador Transformador, o número de descrições textuais fornecidas por imagem e uma função de *data augmentation* para imagens.

Output: Instância do modelo generativo de descrições de imagens.

```
1  Class ImageCaptioningModel () : keras.Model
2      import tensorflow as tf
3      from tensorflow import keras
4      Constructor (cnn_model, encoder, decoder, num_captions_per_image, image_aug)
5          cnn_model ← cnn_model
6          encoder ← encoder
7          decoder ← decoder
8          num_captions_per_image ← num_captions_per_image
9          image_aug ← image_aug
10         loss_tracker ← keras.metrics.Mean(name ← 'loss')
11         acc_tracker ← keras.metrics.Mean(name ← 'accuracy')
12     end
13 end
```

Na classe *Image Captioning Model* também é implementado dois métodos auxiliares para calcular a perda e a acurácia das descrições geradas pelo modelo generativo e um método que as retorna. No Algoritmo 18, esses 3 métodos são implementados.

O método *calculate_loss* recebe como entrada as descrições verdadeiras (*y_true*), as descrições preditas (*y_pred*) e uma máscara que é usada para ignorar os *tokens* de preenchimento (*mask*). A função retorna a perda média considerando apenas as posições de *tokens* relevantes, ou seja, excluindo os *tokens* de preenchimento.

Já o método *calculate_accuracy* calcula a acurácia do modelo, comparando a descrição verdadeira com a descrição predita em cada posição, considerando apenas as posições dos *tokens* relevantes.

O método *metrics* retorna a lista de métricas que serão usadas para avaliar o modelo durante o treinamento e teste. As métricas são atualizadas a cada etapa de treinamento e teste.

Algoritmo 18: Classe *Image Captioning Model* - Métricas

Input: Descrições verdadeiras, descrições preditas e uma máscara.

Output: Métricas de perda e acurácia.

```
1  Class ImageCaptioningModel () : keras.Model
2      import tensorflow as tf
3      calculate_loss (y_true, y_pred, mask)
4          loss ← loss(y_true, y_pred)
5          mask ← tf.cast(mask, dtype ← loss.dtype)
6          loss ← loss * mask
7          loss_tracker ← tf.reduce_sum(loss) / tf.reduce_sum(mask)
8          return loss_tracker
9      end
10     calculate_accuracy (y_true, y_pred, mask)
11         accuracy ← tf.equal(y_true, tf.argmax(y_pred, axis ← 2))
12         accuracy ← tf.math.logical_and(mask, accuracy)
13         accuracy ← tf.cast(accuracy, dtype ← tf.float32)
14         mask ← tf.cast(mask, dtype ← tf.float32)
15         acc_tracker ← tf.reduce_sum(accuracy) / tf.reduce_sum(mask)
16         return acc_tracker
17     end
18     metrics ()
19         return [loss_tracker, acc_tracker]
20     end
21 end
```

Nessa classe também é implementado o método *compute_caption_loss_and_acc*, que pode ser visto no Algoritmo 19. Esse método recebe como entrada os atributos extraídos da imagem (*img_embed*), a sequência de descrição textual fornecida (*batch_seq*) e um argumento lógico (*training*) que indica se o modelo está em modo de treinamento ou não.

Inicialmente, o método chama o codificador Transformador para obter a representação vetorial da imagem com mais valor ao modelo (*encoder_out*). Em seguida, o método separa a sequência de entrada (*batch_seq_inp*) e a sequência verdadeira (*batch_seq_true*) da sequencia fornecida (*batch_seq*). A sequência de entrada é a descrição que será fornecida como entrada

para o modelo de geração de descrição, enquanto que a sequência verdadeira é a descrição correta correspondente à imagem.

O método então cria uma máscara (*mask*) para filtrar os *tokens* de preenchimento e, em seguida, passa a sequência de entrada (*batch_seq_inp*) e a representação vetorial da imagem (*encoder_out*) para um decodificador Transformador, responsável por gerar a descrição da imagem fornecida.

Então, são chamados os métodos *calculate_loss* e *calculate_accuracy* para calcular a perda e acurácia da descrição predita que, em seguida, são retornados.

Algoritmo 19: Classe *Image Captioning Model* - Calcula Métricas

Input: *img_embed*: Atributos extraídos da imagem; *batch_seq*: Sequência de descrição textual fornecida; *training*: Argumento lógico que indica se o modelo está em modo de treinamento ou não.

Output: Métricas de Perda e Acurácia.

```
1  Class ImageCaptioningModel () : keras.Model
2      compute_caption_loss_and_acc (img_embed, batch_seq, training ← True)
3          encoder_out ← encoder(img_embed, training ← training)
4          batch_seq_inp ← batch_seq[:, :-1]
5          batch_seq_true ← batch_seq[:, 1:]
6          mask ← tf.math.not_equal(batch_seq_true, 0)
7          batch_seq_pred ← decoder(batch_seq_inp, encoder_out, training ← training,
             mask ← mask)
8          loss ← calculate_loss(batch_seq_true, batch_seq_pred, mask)
9          acc ← calculate_accuracy(batch_seq_true, batch_seq_pred, mask)
10         return loss, acc
11     end
12 end
```

Nessa classe também é implementado o método *train_step* que é chamado pelo método *fit* do *TensorFlow* durante o treinamento do modelo. Esse método é chamado a cada passo de treinamento do modelo e recebe como entrada um lote de dados que contém imagens médicas e suas respectivas descrições. O método primeiramente aplica uma técnica de aumento de dados *image_augmentation* nas imagens, definido no Algoritmo 20, caso sua utilização esteja habilitada. Esse código retorna um modelo de sequência do *TensorFlow* que realiza três operações nas imagens com o intuito de aumentar a diversidade dos dados de treinamento,

melhorando o desempenho do modelo em dados de teste. As operações são:

a) *RandomFlip('horizontal')*: Faz uma inversão horizontal aleatória na imagem;

b) *RandomRotation(0.2)*: Gira a imagem aleatoriamente em um ângulo de até 0,2 radianos;

c) *RandomContrast(0.3)*: Ajusta o contraste da imagem de forma aleatória com um fator de até 0,3.

Algoritmo 20: Classe *Image Captioning Model* - Aumento de Dados de Imagens

Input: Imagens.

Output: Variações dessas imagens.

```
1  Class ImageCaptioningModel () : keras.Model
2      from tensorflow import keras
3      from tensorflow.keras import layers
4      image_augmentation ()
5          image_augmentation = keras.Sequential([
6              layers.RandomFlip('horizontal'),
7              layers.RandomRotation(0.2),
8              layers.RandomContrast(0.3)
9          ])
10         return image_augmentation
11     end
12 end
```

Aplicado o aumento de dados nas imagens, caso habilitado, o método *train_step* extrai os atributos das imagens usando o CNN e chama o método *compute_caption_loss_and_acc* para calcular a perda e a acurácia para cada descrição do lote, correspondente a cada imagem. Em seguida, o método acumula os valores da perda e acurácia para cada descrição do lote.

Após a atualização desses valores, o método obtém a lista de todos os pesos treináveis do modelo *(train_vars)*, calcula os gradientes da perda em relação a esses pesos, usando o objeto *GradientTape* do *TensorFlow*, e aplica os gradientes aos pesos para atualizar os pesos do modelo usando o otimizador passado no momento da compilação do modelo, o qual entraremos em detalhes mais adiante.

Por fim, o método atualiza os valores das métricas de perda e acurácia para o lote e os retorna. O método *tain_step* pode ser consultado no Algoritmo 21.

Algoritmo 21: Classe *Image Captioning Model* - Passo de Treinamento

Input: Um lote de dados que contém imagens médicas e suas descrições.

Output: Pesos do modelo de descrição de imagens médicas incrementalmente treinados.

```
Class ImageCaptioningModel () : keras.Model
    import tensorflow as tf
    train_step (batch_data)
        batch_img, batch_seq ← batch_data
        batch_loss ← 0
        batch_acc ← 0
        batch_img ← image_augmentation(batch_img)        # Caso habilitado
        img_embed ← cnn_model(batch_img)
        for i in range(num_captions_per_image) do
            loss, acc ← compute_caption_loss_and_acc(img_embed, batch_seq[:, i, :],
              training ← True)
            batch_loss ← batch_loss + loss
            batch_acc ← batch_acc + acc
            train_vars ← (encoder.trainable_variables + decoder.trainable_variables)
            grads ← tf.GradientTape().gradient(loss, train_vars)
            optimizer.apply_gradients(zip(grads, train_vars))
        end
        batch_acc ← batch_acc / float(num_captions_per_image)
        loss_tracker.update_state(batch_loss)
        acc_tracker.update_state(batch_acc)
        return 'loss': loss_tracker.result(), 'acc': acc_tracker.result()
    end
end
```

Há também o método *test_step* que é semelhante ao *train_step*, porém é utilizado para avaliar o desempenho do modelo em um conjunto de dados de teste. Ele processa cada lote de dados de entrada, calculando a perda e a precisão para cada descrição no lote e os respectivos valores acumulados. Entretanto, esse método não atualiza os parâmetros (pesos) do modelo de geração de descrições e não possui aumento de dados das imagens. Esse método pode ser visto no Algoritmo 22.

Algoritmo 22: Classe *Image Captioning Model* - Passo de Teste

Input: Um lote de dados que contém imagens médicas e suas descrições.

Output: Métricas de perda e precisão do lote passado.

```
Class ImageCaptioningModel () : keras.Model
    test_step (batch_data)
        batch_img, batch_seq ← batch_data
        batch_loss ← 0
        batch_acc ← 0
        img_embed ← cnn_model(batch_img)
        for i in range(num_captions_per_image) do
            loss, acc ← compute_caption_loss_and_acc(img_embed, batch_seq[:, i, :],
              training ← False)
            batch_loss ← batch_loss + loss
            batch_acc ← batch_acc + acc
        end
        batch_acc ← batch_acc / float(num_captions_per_image)
        loss_tracker.update_state(batch_loss)
        acc_tracker.update_state(batch_acc)
        return 'loss': loss_tracker.result(), 'acc': acc_tracker.result()
    end
end
```

Em resumo, essa classe implementa um modelo generativo de descrição de imagens médicas que usa uma CNN para extrair os atributos das imagens, um codificador Transformador para codificar os atributos das imagens em uma representação mais relevante ao modelo e um decodificador Transformador para gerar as descrições da imagens. O modelo é treinado usando um dos algoritmo de otimização de gradiente estudados para minimizar a perda da descrição gerada em relação à descrição verdadeira. Durante o treinamento, os dados de imagem são aumentados usando a sequência de transformação definida pela função *image_augmentation()*, caso habilitado, e as métricas de perda e acurácia são calculadas e atualizadas a cada etapa de treinamento e podem ser acessadas por meio do método *metrics* da classe.

No Algoritmo 23, é instanciado um modelo generativo de descrição de imagens médicas, combinando tudo que foi abordado nessa seção. Vale lembrar que as constantes foram

inicializadas no Algoritmo 7.

Algoritmo 23: Instanciação do Modelo Generativo de Descrição de Imagens Médicas

Input: CNN, Codificador Transformador, Decodificador Transformador e função de aumento de dados de imagem.

Output: Instância do modelo generativo de descrição de imagens médicas.

```
1  GetImageCaptioningModel ()
2      cnn_model ← get_cnn_model()
3      encoder ← TransformerEncoderBlock(
4          embed_dim ← EMBED_DIM,
5          num_heads ← ENCODER_NUM_HEADS
6      )
7      decoder ← TransformerDecoderBlock(
8              embed_dim ← EMBED_DIM,
9              ff_dim ← FF_DIM,
10             num_heads ← DECODER_NUM_HEADS
11     )
12     caption_model ← ImageCaptioningModel(
13             cnn_model ← cnn_model,
14             encoder ← encoder,
15             decoder ← decoder,
16             num_captions_per_image ← NUM_CAPTIONS_PER_IMAGE,
17             image_aug ← image_augmentation()
18     )
19     return caption_model
20 end
```

3.4.6 Treinamento do Modelo

Com o modelo generativo instanciado, é necessário realizar seu treinamento com os dados previamente preparados para que ele seja capaz de produzir as descrições de imagens médicas.

Para isso, o primeiro passo é a definição de uma função de perda (*cross_entropy*) usando a classe *SparseCategoricalCrossentropy* do *TensorFlow* que calcula a entropia cruzada

entre as previsões do modelo e os valores verdadeiros.

Em seguida, é definido um critério de parada do treinamento usando a classe *EarlyStopping* do *TensorFlow* que interrompe o treinamento se a performance do modelo na validação não melhorar por um número de épocas especificado no parâmetro *patience*. Esse critério de parada também é responsável por retornar os melhores pesos do modelo com base na validação.

Esses 2 primeiros passos estão representados no Algoritmo 24.

Algoritmo 24: Definição da Função de Perda e Critério de Parada do Treinamento

Input: Objetos *SparseCategoricalCrossentropy* e *EarlyStopping* do *TensorFlow*.

Output: Função de perda e critério de parada do treinamento do modelo generativo definidos.

```
1  ModelTranning ()
2      import tensorflow as tf
3      from tensorflow import keras
4      cross_entropy ← keras.losses.SparseCategoricalCrossentropy(
5          from_logits ← False,
6          reduction ← 'none'
7      )
8      early_stopping ← keras.callbacks.EarlyStopping(
9          patience ← 3,
10         restore_best_weights ← True
11     )
12     return cross_entropy, early_stopping
13 end
```

O próximo passo é definir uma taxa de aprendizagem para o modelo generativo. Para isso, implementou-se uma classe *LRSchedule*, que pode ser consultada no Algoritmo 25, que herda a classe *LearningRateSchedule* do *TensorFlow*. Essa classe implementa um gerenciador de taxa de aprendizagem que começa com um valor baixo e aumenta gradualmente até um valor definido, seguindo uma estratégia conhecida como *warm-up*. O valor máximo da taxa de aprendizagem é definido no parâmetro *post_warmup_learning_rate*, enquanto o número de passos para a etapa de aquecimento é definido no parâmetro *warmup_steps*. Portanto, a taxa de aprendizagem é atualizada em cada passo do treinamento, aumentando gradualmente durante a etapa de aquecimento e se mantendo constante após isso.

Algoritmo 25: Definição da Taxa de Aprendizagem do Modelo

Input: Objeto *LearningRateSchedule* do *TensorFlow*.

Output: Taxa de aprendizagem do modelo generativo definida.

```
1  ModelTranning ()
2      import tensorflow as tf
3      from tensorflow import keras
4      Class LRSchedule () : LearningRateSchedule
5          Constructor (post_warmup_learning_rate, warmup_steps)
6              post_warmup_learning_rate ← post_warmup_learning_rate
7              warmup_steps ← warmup_steps
8          end
9          Call (step)
10             global_step ← tf.cast(step, tf.float32)
11             warmup_steps ← tf.cast(self.warmup_steps, tf.float32)
12             warmup_progress ← global_step / warmup_steps
13             warmup_learning_rate ← post_warmup_learning_rate * warmup_progress
14             tf_cond ← tf.cond(
15                 global_step < warmup_steps,
16                 lambda: warmup_learning_rate,
17                 lambda: post_warmup_learning_rate
18             )
19             return tf_cond
20         end
21 end
```

A última parte do treinamento do modelo generativo pode ser consultada no Algoritmo 26. Com a classe *LRSchedule* definida, o próximo passo é definir o número de passos de treino (*num_train_steps*) e o número de passos de aquecimento da taxa de aprendizagem (*num_warmup_steps*). Então, é instanciado um objeto gerenciador de taxa de aprendizagem (*lr_schedule*). Em seguida, o modelo generativo instanciado no Algoritmo 23 é compilado utilizando a função de perda *cross_entropy* e um dos otimizadores de gradiente estudados com a taxa de aprendizagem definida pelo objeto *lr_schedule*. Por fim, o modelo é efetivamente treinado utilizando o método *fit*, passando o conjunto de treinamento *train_dataset*, a proporção

do conjunto de dados de treino a ser usado na validação *validation_split*, o número de épocas *EPOCHS* (definido no Algoritmo 7) e a lista de *callbacks* que inclui o critério de parada de treinamento *early_stopping*.

Algoritmo 26: Compilação e Treinamento do Modelo

Input: Instância do modelo generativo, conjunto de dados de treinamento, quantidade de épocas de treinamento, taxa de aprendizagem máxima, otimizador de gradiente, função de perda e a condição de parada do treinamento.

Output: Modelo generativo treinado.

```
1  ModelTranning ()
2      import tensorflow as tf
3      from tensorflow import keras
4      num_train_steps ← len(train_dataset) * EPOCHS
5      num_warmup_steps ← num_train_steps // 15
6      lr_schedule ← LRSchedule(
7          post_warmup_learning_rate ← 1e-4,
8          warmup_steps ← num_warmup_steps
9      )
10     caption_model.compile(
11         optimizer ← <Otimizador Analisado>(lr_schedule),
12         loss ← cross_entropy
13     )
14     caption_model.fit(
15         train_dataset,
16         validation_split ← 0.2,
17         epochs ← EPOCHS,
18         callbacks ← [early_stopping]
19     )
20     return caption_model
21 end
```

Resumidamente, durante o treinamento, o modelo ajusta seus parâmetros para minimizar a função de perda *cross_entropy*, utilizando um dos otimizadores de gradiente estudados, com a taxa de aprendizagem ajustada pelo gerenciador *lr_schedule*. Ao final dessa etapa, a expectativa é ter um modelo generativo de descrição de imagens médicas eficaz. Para determinar

se esse modelo generativo é de fato útil, o próximo passo é gerar diversas descrições de um conjunto de teste e avaliar os resultados.

3.4.7 Geração de Descrições de Imagens Médicas

Para gerar as descrições das imagens médicas do conjunto de teste utilizando o modelo generativo treinado na etapa anterior, implementou-se a classe *GeracaoDescricoesImagens*.

A primeira etapa, descrita no Algoritmo 27 é definir algumas variáveis e importações que serão usadas posteriormente. A variável *vocab* contém o vocabulário usado pelo modelo e é obtida através de uma chamada à função *get_vocabulary()* do objeto *vectorization*, construído no Algoritmo 10. A variável *index_lookup* é um dicionário que atribui índices as palavras do vocabulário. A variável *max_decoded_sentence_length* é o comprimento máximo permitido para a descrição gerada. Finalmente, a variável *test_images* é uma lista que contém os nomes das imagens do conjunto de teste.

Algoritmo 27: Classe Geração das Descrições das Imagens Médicas - Inicializações

Input: Conjunto de dados de teste.

Output: Descrições das imagens do conjunto de teste.

```
1 Class GeracaoDescricoesImagens ()
2     import numpy as np
3     import matplotlib.pyplot as pl
4     import tensorflow as tf
5     vocab ← vectorization.get_vocabulary()
6     index_lookup ← dict(zip(range(len(vocab)), vocab))
7     max_decoded_sentence_length ← SEQ_LENGTH - 1
8     test_images ← list(test_data.keys())
9 end
```

A segunda etapa, descrita no Algoritmo 28, é definir uma função chamada *generate_caption* dentro da classe *GeracaoDescricoesImagens* que recebe um índice de imagem como entrada e usa o modelo treinado anteriormente para gerar uma descrição para a imagem correspondente.

Algoritmo 28: Classe Geração das Descrições das Imagens Médicas - *generate_caption*

Input: Conjunto de dados de teste.

Output: Descrições das imagens do conjunto de teste.

```
Class GeracaoDescricoesImagens ()
    generate_caption (image_index)
        sample_img ← test_images[image_index]
        sample_img_name ← sample_img
        sample_img ← decode_and_resize(sample_img)
        img ← sample_img.numpy().clip(0, 255).astype(np.uint8)
        plt.imshow(img)
        plt.show()
        img ← tf.expand_dims(sample_img, 0)
        img ← caption_model.cnn_model(img)
        encoded_img ← caption_model.encoder(img, training ← False)
        decoded_caption ← '<start> '
        for i in range(max_decoded_sentence_length) do
            tokenized_caption ← vectorization([decoded_caption])[:, :-1]
            mask ← tf.math.not_equal(tokenized_caption, 0)
            predictions ← caption_model.decoder(tokenized_caption, encoded_img,
              training ← False, mask ← mask)
            sampled_token_index ← np.argmax(predictions[0, i, :])
            sampled_token ← index_lookup[sampled_token_index]
            if sampled_token == ' <end>' then
                break
            end
            decoded_caption ← decoded_caption + ' ' + sampled_token
        end
        decoded_caption ← decoded_caption.replace('<start> ', '')
        decoded_caption ← decoded_caption.replace(' <end>', '').strip()
        print('Imagem: ' + sample_img_name)
        print('Predicted Caption: ', decoded_caption)
        print('Label: ', captions_mapping.get(sample_img_name), '\n')
    end
end
```

Dentro da função *generate_caption*, a imagem é lida do disco e redimensionada para um tamanho específico usando a função *decode_and_resize*, definida no Algoritmo 11. A imagem redimensionada é exibida usando a biblioteca *matplotlib*.

Em seguida, a imagem é passada para a CNN do modelo para extração de características, definido no Algoritmo 12. Posteriormente, essas características são passados ao codificador Transformador, definido no Algoritmo 13, que gera uma representação vetorial das características da imagem com valor ao modelo. Por fim, a descrição da imagem é construída através do decodificador Transformador, definido nos Algoritmos 14, 15 e 16.

Entrando em mais detalhes na aplicação do decodificador Transformador, a descrição da imagem é construída iterativamente em um laço de repetição, adicionando uma palavra de cada vez. Dentro do laço, a descrição construída até o momento é transformada em *token* usando a função *vectorization* e uma máscara é criada para garantir que o decodificador Transformador não processe palavras que ainda não foram geradas. Em seguida, o decodificador Transformador é usado para calcular a probabilidade de ocupação das palavras do vocabulário na próxima posição da descrição. A próxima palavra da descrição é selecionada como a palavra de maior probabilidade calculada. O laço continua até que a palavra <*end*> seja gerada pelo modelo ou até que o comprimento máximo da descrição seja alcançado.

Depois que a descrição é gerada, ela é processada para remover as palavras <*start*> e <*end*> e quaisquer espaços em branco desnecessários. Em seguida, a descrição gerada é impressa juntamente com o nome da imagem e a descrição real correspondente, obtida pelo mapeamento do nome da imagem e sua respectiva descrição verdadeira existente no objeto *captions_mapping*, gerada no Algoritmo 8.

Finalmente, no Algoritmo 29, a função *generate_caption* é executada para cada imagem do conjunto de teste da lista *test_images*. Com isso, comparando-se as descrições geradas com as descrições verdadeiras, é possível avaliar a capacidade do modelo de gerar descrições coerentes.

Algoritmo 29: Classe Geração das Descrições das Imagens Médicas - Geração

Input: Conjunto de dados de teste.

Output: Descrições das imagens do conjunto de teste.

```
1 Class GeracaoDescricoesImagens ()
2     for image_index in range(0, len(test_images)) do
3         generate_caption(image_index)
4     end
5 end
```

3.5 Avaliação dos Resultados

Nesta seção, apresenta-se as métricas utilizadas para avaliar o desempenho do modelo proposto, bem como os tempos de processamento envolvidos durante o treinamento e teste. Além disso, fornece-se detalhes sobre a configuração do computador utilizado para executar os experimentos.

3.5.1 Métricas de Avaliação de Desempenho

Para avaliar o desempenho do modelo, utilizou-se 2 métricas principais: acurácia e perda. A acurácia mensura a proporção de palavras corretas geradas em relação ao total de palavras geradas e pode ser calculada pela equação

$$Acc = \frac{\text{Número de palavras corretas geradas}}{\text{Número total de palavras geradas}}. \tag{3.1}$$

Já a perda representa a medida do erro entre as previsões das palavras do modelo e as palavras reais do conjunto de dados. Neste trabalho, a perda é calculada pela Entropia Cruzada Categórica Esparsa. Essa função é comumente utilizada em tarefas de classificação multi-classe, onde os rótulos verdadeiros são representados por valores inteiros, ou seja por uma codificação esparsa, e as probabilidades previstas pelo modelo são calculadas usando uma função de ativação *softmax* para obter uma distribuição de probabilidade sobre as classes (CHOLLET, 2021). A Entropia Cruzada Categórica Esparsa busca minimizar a diferença entre as probabilidades previstas pelo modelo e os rótulos verdadeiros, penalizando mais fortemente as classificações incorretas e pode ser calculada pela equação

$$\text{Erro}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \ln(\hat{y}_i). \tag{3.2}$$

3.5.2 Métricas de Avaliação do Processamento

Para avaliar o processamento do modelo, mensurou-se o tempo de treinamento e o tempo de teste. O tempo de treinamento é o tempo necessário para treinar o modelo com o conjunto de dados de treinamento, enquanto o tempo de teste é o tempo necessário para realizar as predições em um conjunto de dados de teste. Essas métricas são medidas em segundos e fornecem uma estimativa do tempo necessário para realizar operações relacionadas ao modelo.

Esses tempos dependem do *hardware* utilizado. Portanto, é importante ressaltar que os experimentos foram conduzidos em um *notebook* Dell Alienware M15 R7 com as seguintes configurações:

a) Processador: Intel Core de 12ª Geração I7-12700H de 2,30 GHz de 20 núcleos;
b) Memória RAM: 32 GB;
c) Armazenamento: HD SSD de 1 TB;
d) Placa de Vídeo: NVIDIA GeForce RTX 3070 Ti com 8 GB de memória dedicada;
e) Sistema Operacional: Windows 11.

3.5.3 Matriz de Confusão

Para o melhor modelo, elaborou-se uma matriz de confusão para cada uma das partes do texto geradas, que incluem o Tipo de Exame, Parte do Corpo e o Problema Identificado.

A matriz de confusão é uma tabela que permite visualizar as predições corretas e incorretas do modelo em relação às classes verdadeiras, além de fornecer várias métricas como *Sensitivity*, *Specificity*, *Precision*, *Negative Predictive Value*, *Accuracy* e *F1 Score* (HEYDARIAN *et al.*, 2022).

Essas métricas são calculadas levando em conta o valor de uma série de variáveis extraídas das matrizes de confusão. Essas variáveis estão listadas logo abaixo e, no contexto deste trabalho, têm os seguintes significados:

a) *Positive* (P): Número total de exemplos de uma dada classe;
b) *Negative* (N): Número total de exemplos de outras classes;
c) *True Positive* (TP): Número de exemplos corretamente classificados como positivos;
d) *False Positive* (FP): Número de exemplos erroneamente classificados como positivos;

e) *True Negative* (TN): Número de exemplos corretamente classificados como negativos;

f) *False Negative* (FN): Número de exemplos erroneamente classificados como negativos;

A seguir, define-se as métricas das matrizes de confusão calculadas neste trabalho.

Sensitivity (Sensibilidade) é a proporção de exemplos positivos corretamente classificados em relação ao total de exemplos positivos e pode ser calculada pela equação

$$\text{Sensibilidade} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}. \tag{3.3}$$

Specificity (Especificidade) é a proporção de exemplos negativos corretamente classificados em relação ao total de exemplos negativos e pode ser calculada pela equação

$$\text{Especificidade} = \frac{\text{Verdadeiros Negativos}}{\text{Verdadeiros Negativos} + \text{Falsos Positivos}}. \tag{3.4}$$

Precision (Precisão) é a proporção de exemplos corretamente classificados como positivos em relação ao total de exemplos classificados como positivos e pode ser calculada pela equação

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}. \tag{3.5}$$

Negative Predictive Value (Valor Preditivo Negativo) é a proporção de exemplos corretamente classificados como negativos em relação ao total de exemplos classificados como negativos e pode ser calculado pela equação

$$\text{Valor Preditivo Negativo} = \frac{\text{Verdadeiros Negativos}}{\text{Verdadeiros Negativos} + \text{Falsos Negativos}}. \tag{3.6}$$

Accuracy (Acurácia) é a proporção geral de exemplos corretamente classificados em relação ao total de exemplos e pode ser calculada pela equação

$$\text{Acc} = \frac{\text{Verdadeiros Positivos} + \text{Verdadeiros Negativos}}{\text{Verdadeiros Positivos} + \text{Verdadeiros Negativos} + \text{Falsos Positivos} + \text{Falsos Negativos}}. \tag{3.7}$$

F1 Score é uma medida que combina a precisão e a sensibilidade em uma única métrica. É a média harmônica entre essas duas métricas e fornece um equilíbrio entre elas e pode ser calculada pela equação

$$\text{F1 Score} = 2 \times \frac{\text{Precisão} \times \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}}. \tag{3.8}$$

Também para o melhor modelo, calculou-se o BLEU-1 e o comparamos com outros modelos mencionados na fundamentação teórica. O BLEU-1 é uma métrica comumente usada para avaliar a qualidade de traduções ou geração de texto (PAPINENI *et al.*, 2001). Ele mensura a sobreposição de unigramas entre as sequências geradas e as sequências de referência e pode ser calculado pela equação

$$\text{BLEU-1} = \frac{\sum_{\text{1-gramas}} \text{número de 1-gramas corretamente preditos}}{\sum_{\text{1-gramas}} \text{número total de 1-gramas preditos}}. \tag{3.9}$$

4 RESULTADOS

Neste capítulo, apresenta-se os resultados obtidos por meio da análise dos dados e experimentos realizados. Com base nas seções anteriores, aprofunda-se a compreensão dos resultados alcançados, descrevendo a geração do conjunto de dados, a determinação do melhor modelo, as análises realizadas sobre esse modelo e, por fim, a comparação do mesmo com a literatura.

4.1 Geração do Conjunto de Dados

O primeiro passo para a implementação de um modelo generativo de descrições de imagens médicas é construir os conjuntos de dados de treino, teste e validação. Para isso, foi necessário delimitar o contexto de um subconjunto de imagens do ROCO utilizando as palavras de interesse definidas anteriormente.

Portanto, as palavras de interesse da Tabela 2 foram inseridas em três listas distintas no Algoritmo 5, o qual criou uma lista de todas as combinações possíveis, tomadas de 3 em 3, dos valores de interesse definidos em cada categoria.

Para cada combinação de palavras de interesse, o Algoritmo 6 procurou imagens que correspondiam a esses valores no dicionário criado pelo Algoritmo 4 e as salvou em um diretório especificado.

Como explicado no capítulo anterior, o Algoritmo 6 iterou sobre a lista de todas as combinações possíveis e separou os grupos de imagens do ROCO pertencentes as combinações das palavras de interesse que continham entre 10 e 80 imagens. Com isso, gerou-se um conjunto de dados contendo 1.419 imagens médicas e seus respectivos textos descritivos. Os textos e a quantidade de imagens de cada grupo podem ser consultadas na Tabela 3.

Então, esse conjunto de 1.419 imagens médicas foi dividido aleatoriamente em três partes menores, na proporção 60%/20%/20%, consistindo nos conjuntos de treino, validação e teste dos modelos generativos implementados neste estudo.

4.2 Determinação do Melhor Modelo

Nessa seção, os resultados serão apresentados e discutidos através de uma série de tabelas. Essas tabelas apresentam os resultados do experimento realizado para criar um modelo generativo de textos descritivos de imagens médicas. Cada linha dessas tabelas representa uma

Tabela 3 – Grupos de Palavras de Interesse Selecionados

Tipo do Exame	Parte do Corpo	Problema Identificado	Quantidade de Imagens
ct	chest	tumor	52
ct	artery	aneurysm	63
ct	abdoman	tumor	38
ct	abdoman	cystic	58
ct	abdoman	hepatic	47
ct	abdoman	pancreatic	35
ct	abdoman	hematoma	38
ct	lobe	hepatic	51
ct	lung	effusion	56
ct	head	pancreatic	56
ct	vein	hepatic	34
ct	liver	cyst	35
ct	liver	hepatic	28
scan	chest	nodule	77
scan	abdoman	cystic	23
scan	lobe	hepatic	18
scan	lung	effusion	18
scan	lung	nodule	41
scan	pulmonary	nodule	17
scan	head	pancreatic	13
tomography	chest	tumor	29
tomography	lobe	nodule	41
tomography	lobe	hepatic	21
tomography	lung	effusion	20
tomography	lung	nodule	34
tomography	pulmonary	nodule	21
radiograph	chest	normal	47
radiograph	bone	fracture	41
radiograph	pelvi	fracture	44
radiograph	neck	fracture	31
radiograph	femoral	fracture	31
radiograph	hip	fracture	32
xray	chest	normal	65
contrast	artery	aneurysm	32
abdominal	aortic	aneurysm	35
angiography	artery	normal	39
angiogram	artery	aneurysm	58
		Total	1419

Fonte: Elaboração própria.

configuração de experimento específica, com diferentes combinações de modelos de extração de atributos de imagens, otimizadores de gradiente, uso de *Transfer Learning* e o uso de aumento de dados.

Para melhor entendimento dos resultados, é válido definir cada uma das colunas das tabelas:

a) *ID*: Identificador único para cada configuração de experimento;

b) *OPTZ*: Otimizador de gradiente utilizado no treinamento do modelo;

c) *TL*: Sigla para *Transfer Learning* que indica se foi utilizado o conceito de *Transfer*

Learning do ImageNet;

d) *TR*: Sigla para *Trainable* que indica se houve ajuste dos pesos durante a etapa de treinamento;

e) *IA*: Sigla para *Image Augmentation* que indica se foi aplicada uma técnica de aumento de dados nas imagens;

f) *Loss*: Valor final da função de perda obtido pelo modelo;

g) *Acc*: Acurácia final alcançada pelo modelo;

h) *Epochs*: Número de épocas de treinamento;

i) *Training Time (s)*: Tempo necessário para o treinamento do modelo em segundos;

j) *Test Time (s)*: Tempo necessário para o teste do modelo em segundos.

Essas métricas fornecem informações sobre o desempenho dos modelos em termos de perda, acurácia e tempo de treinamento/teste. Com base nessas informações, é possível comparar diferentes configurações de experimento e identificar aquelas que apresentaram melhores resultados.

Essas informações permitem avaliar o desempenho dos modelos em relação às diferentes combinações de variáveis testadas, como extrator de atributos, otimizadores de gradiente, o uso do *Transfer Learning* e técnicas de aumento de dados.

Vale ressaltar que, após a realização de todos os 192 experimentos, foi calculada a média (μ) e o desvio padrão (σ) da acurácia, obtendo-se os valores apresentados na Tabela 4. Esses resultados indicam que os experimentos apresentam diferenças estatisticamente significativas em relação à acurácia. Portanto, levando-se em conta apenas essa métrica, é possível definir o melhor modelo como aquele que tem a maior acurácia.

Tabela 4 – Média e Desvio Padrão da Acurária

CNN	μ	σ	**Mín**	**Máx**	$\mu - \sigma$	$\mu + \sigma$
DenseNet201	0,7365	0,0114	0,7151	0,7610	0,7250	0,7479
ResNet152V2	0,7154	0,0138	0,6868	0,7359	0,7016	0,7292
NASNetLarge	0,7050	0,0085	0,6891	0,7210	0,6965	0,7135
VGG19	0,7431	0,0167	0,7056	0,7628	0,7264	0,7598
Xception	0,7313	0,0116	0,7069	0,7499	0,7197	0,7429
InceptionV3	0,7236	0,0124	0,6992	0,7472	0,7111	0,7360
InceptionResNetV2	0,6985	0,0089	0,6823	0,7165	0,6896	0,7074
MobileNetV2	0,7056	0,0225	0,6603	0,7345	0,6831	0,7281
Global	0,7199	0,0206	0,6603	0,7628	0,6992	0,7404

Fonte: Elaboração própria.

4.2.1 Família DenseNet201

Ao avaliar os resultados da Tabela 5, levando em conta as diferentes variáveis presentes na tabela, podemos identificar algumas tendências relevantes.

Tabela 5 – Resultados da Família DenseNet201

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
1	Adam		S		0,7487	0,7276	20	6520	246
2	AdamW		S		0,7727	0,7205	6	1656	250
3	Adadelta		S		0,7601	0,7253	8	2185	248
4	Adafactor		S		0,7539	0,7275	7	1896	259
5	Adam		S	S	0,7894	0,7151	20	5662	249
6	AdamW		S	S	0,7461	0,7271	6	1706	281
7	Adadelta		S	S	0,7390	0,7279	6	1784	293
8	Adafactor		S	S	0,7199	0,7356	6	1812	314
9	Adam	S			0,7438	0,7443	10	2887	293
10	AdamW	S			0,7244	0,7406	4	1160	298
11	Adadelta	S			0,7223	0,7413	8	2345	306
12	Adafactor	S			0,7190	0,7446	5	1476	319
13	Adam	S		S	0,6907	0,7413	18	5463	334
14	AdamW	S		S	0,6890	0,7483	4	1454	400
15	Adadelta	S		S	0,6818	0,7549	10	3400	357
16	Adafactor	S		S	0,6807	0,7532	7	3010	443
17	Adam	S	S		0,7519	0,7337	11	4778	345
18	AdamW	S	S		0,7534	0,7439	4	1203	329
19	Adadelta	S	S		0,7408	0,7449	5	1540	346
20	Adafactor	S	S		0,7190	**0,7610**	6	2205	420
21	Adam	S	S	S	0,7236	0,7262	18	6640	424
22	AdamW	S	S	S	0,7349	0,7242	4	1530	436
23	Adadelta	S	S	S	0,7059	0,7343	8	2733	366
24	Adafactor	S	S	S	0,7044	0,7318	4	1649	433
							Total	66694	7989

Fonte: Elaboração própria.

Primeiro, os diferentes otimizadores de gradiente têm um impacto considerável no desempenho do modelo. O Adafactor parece ser eficaz, mostrando as menores perdas e as acurácias mais altas em várias configurações. Isso sugere que o Adafactor pode ser um otimizador eficaz para o treinamento do modelo DenseNet201 nesse contexto específico.

Além disso, em geral, a utilização do *Transfer Learning* melhora os resultados em relação às configurações sem *Transfer Learning*. As configurações com TL alcançam perdas mais baixas e acurácias mais altas em várias combinações de otimizadores de gradiente. Isso destaca a importância de aproveitar os conhecimentos pré-treinados de modelos mais amplos e adaptá-los para tarefas específicas.

É importante destacar que os melhores resultados dessa família ocorreram nos casos onde foi possível ajustar os pesos do modelo durante a etapa de treinamento. Isso significa que *ImageNet* pode até ser um bom ponto de partida desses pesos, mas resultados melhores podem ser alcançados quando se faz um ajuste desses pesos durante a etapa de treinamento.

Quanto a utilização de aumento de dados, as configurações com *Image Augmentation*

geralmente apresentaram perdas mais baixas e acurácias mais altas, sugerindo que o aumento de dados foi benéfico para o desempenho do modelo. Percebeu-se que essa técnica pode melhorar a generalização do modelo, aumentando a robustez e reduzindo o *overfitting*.

Por fim, as configurações apresentam variações significativas nos tempos de treinamento, que vão desde algumas centenas de segundos até várias horas. Essas diferenças podem ser influenciadas por vários fatores, como o tamanho do conjunto de dados, a complexidade do modelo, entre outros. Em geral, as configurações com *Transfer Learning* e técnicas de *Image Augmentation* tendem a exigir mais tempo de treinamento devido ao processamento adicional necessário para ajustar os pesos do modelo ou aplicar transformações nos dados. Por outro lado, as configurações sem *Transfer Learning* e sem *Image Augmentation* podem ter tempos de treinamento mais curtos, pois estão lidando apenas com o conjunto de dados original e não realizam ajustes complexos no modelo.

Ao avaliar os resultados das tabelas, podemos considerar vários fatores para determinar o melhor resultado. A métrica de acurácia (Acc) é um indicador importante do desempenho do modelo, pois mensura a proporção de predições corretas em relação ao total de predições. Além disso, podemos levar em conta outras métricas, como a perda (Loss), a medida F1 e os indicadores de precisão e taxa de acerto.

Analisando os resultados, podemos observar que algumas configurações apresentam uma acurácia relativamente alta. Por exemplo, a configuração de ID 20 obteve uma acurácia de 0,7610, que é a mais alta entre as configurações testadas. Essa configuração utilizou a arquitetura DenseNet201 como extrator de atributos de imagens médicas, o otimizador Adafactor, *Transfer Learning* com ajuste dos pesos e *Image Augmentation*.

Além disso, é preciso levar em conta outros critérios, como o tempo de processamento e a escalabilidade do modelo. Algumas configurações que alcançaram uma alta acurácia podem exigir um tempo de treinamento significativamente maior, o que pode ser impraticável em alguns casos. Portanto, é importante encontrar um equilíbrio entre o desempenho e o tempo de processamento, considerando os recursos disponíveis e as restrições do projeto.

Com base nos resultados apresentados, a configuração de ID 20 parece ser uma das mais promissoras, obtendo uma alta acurácia e envolvendo técnicas como *Transfer Learning*, *Image Augmentation* e ajuste de pesos.

4.2.2 Família ResNet152V2

Ao avaliar os resultados da Tabela 6, levando em conta as diferentes variáveis presentes na tabela, podemos identificar algumas tendências relevantes.

Tabela 6 – Resultados da Família ResNet152V2

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
25	Adam		S		0,8128	0,7020	20	7844	293
26	AdamW		S		0,8027	0,7113	4	1654	385
27	Adadelta		S		0,7894	0,7174	10	3999	294
28	Adafactor		S		0,7840	0,7127	7	2815	308
29	Adam		S	S	0,8791	0,6868	11	4493	299
30	AdamW		S	S	0,8525	0,6929	5	2059	332
31	Adadelta		S	S	0,8480	0,6908	10	4116	355
32	Adafactor		S	S	0,8499	0,6919	6	2428	337
33	Adam	S			0,7817	0,7277	15	5974	338
34	AdamW	S			0,7539	0,7278	11	4395	318
35	Adadelta	S			0,7524	0,7263	7	2912	366
36	Adafactor	S			0,7413	**0,7359**	9	3771	452
37	Adam	S		S	0,7865	0,7151	18	7519	382
38	AdamW	S		S	0,7712	0,7248	6	2548	372
39	Adadelta	S		S	0,7722	0,7228	7	2957	361
40	Adafactor	S		S	0,7533	0,7311	11	4672	384
41	Adam	S	S		0,7424	0,7237	19	7932	381
42	AdamW	S	S		0,7593	0,7124	4	1647	334
43	Adadelta	S	S		0,7469	0,7136	7	2891	359
44	Adafactor	S	S		0,7271	0,7246	8	3645	1042
45	Adam	S	S	S	0,7815	0,7033	20	9948	396
46	AdamW	S	S	S	0,7629	0,7214	4	1648	339
47	Adadelta	S	S	S	0,7523	0,7234	6	2713	453
48	Adafactor	S	S	S	0,7425	0,7298	11	5785	489
							Total	100363	9371

Fonte: Elaboração própria.

Em relação aos otimizadores, o otimizador AdamW (IDs 26, 30, 34, 38) tende a apresentar bons resultados em termos de acurácia, com valores acima de 0,72, e perdas relativamente baixas. O otimizador Adafactor (linhas 28, 32, 36, 40) também mostra resultados promissores em termos de acurácia, com valores acima de 0,73.

A utilização de técnicas de transferência de aprendizado (TL) e aumento de dados (IA) pode levar a melhorias significativas no desempenho do modelo. Observa-se que várias configurações que aplicam essas técnicas apresentam uma acurácia mais alta em comparação com as configurações sem essas técnicas. Por exemplo, as linhas 29, 33, 37, 41 mostram resultados superiores quando tanto a transferência de aprendizado quanto o aumento de dados são aplicados.

É interessante notar que, em algumas configurações, a inclusão de transferência de aprendizado e/ou aumento de dados pode levar a um aumento no tempo de treinamento. Isso ocorre porque essas técnicas geralmente envolvem o uso de modelos pré-treinados ou a geração de mais exemplos de treinamento, o que requer mais tempo de processamento. Portanto, ao considerar essas técnicas, é necessário levar em conta o equilíbrio entre o desempenho do

modelo e o tempo de treinamento necessário. No entanto, o tempo de teste não parece ser afetado significativamente pelas configurações avaliadas.

Os resultados indicam que o uso de otimizadores específicos, como Adafactor e AdamW, juntamente com a transferência de aprendizado, pode levar a um desempenho promissor em termos de acurácia.

O melhor resultado da Tabela 6, considerando a maior acurácia alcançada, é o da configuração de ID 36. Essa configuração obteve uma acurácia de 0,7359 após 9 épocas de treinamento.

Nessa configuração, foi utilizado o modelo ResNet152V2 com o otimizador Adafactor e a transferência de aprendizado (TL) foi aplicada.

Além disso, é relevante considerar o tempo de treinamento e teste associado a essa configuração. Na Tabela 6, é indicado que o tempo de treinamento foi de 3771 segundos (aproximadamente 1 hora e 3 minutos) e o tempo de teste foi de 452 segundos (pouco menos de 8 minutos). Portanto, é necessário levar em conta a acurácia alcançada juntamente com o tempo de treinamento e teste para avaliar a eficiência e a praticidade dessa configuração em um contexto real.

4.2.3 Família NASNetLarge

Na Tabela 7, é apresentada uma série de experimentos realizados com o modelo NASNetLarge, onde diferentes combinações das variáveis foram exploradas.

Foram testados quatro otimizadores durante o treinamento dos modelos: Adam, AdamW, Adadelta e Adafactor. Cada otimizador possui suas próprias características de ajuste de pesos e taxas de aprendizado. Os resultados mostram que não há uma diferença significativa na acurácia alcançada pelos diferentes otimizadores, variando de aproximadamente 0,69 a 0,72.

Quanto ao uso de *Transfer Learning* pré-treinado com o conjunto de dados ImageNet, os resultados indicam que o seu uso tende proporcionar acurácias um pouco melhores em comparação com os experimentos sem *Transfer Learning*. Isso sugere que a transferência de conhecimento de modelos pré-treinados pode auxiliar no aprendizado de características relevantes para imagens médicas.

Além do uso de *Transfer Learning*, foi avaliado o ajuste dos pesos durante o treinamento. Os resultados mostram que o ajuste dos pesos também pode contribuir para um desempenho ligeiramente melhor, com acurácias um pouco mais altas em comparação com os

Tabela 7 – Resultados da Família NASNetLarge

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
49	Adam		S		0,8557	0,6914	16	7587	466
50	AdamW		S		0,8530	0,6980	5	2427	479
51	Adadelta		S		0,8526	0,6900	5	2418	495
52	Adafactor		S		0,8356	0,7011	8	3887	482
53	Adam		S	S	0,8414	0,7001	18	8774	490
54	AdamW		S	S	0,8509	0,6891	4	2005	551
55	Adadelta		S	S	0,8475	0,6969	8	3699	537
56	Adafactor		S	S	0,8459	0,6949	5	2494	523
57	Adam	S			0,8125	0,7084	11	5433	531
58	AdamW	S			0,8115	0,7095	4	1967	537
59	Adadelta	S			0,7938	0,7105	7	3798	674
60	Adafactor	S			0,7849	0,7173	6	3864	765
61	Adam	S		S	0,8122	0,7065	15	10067	822
62	AdamW	S		S	0,7857	0,7089	6	3495	707
63	Adadelta	S		S	0,7780	0,7137	6	4172	832
64	Adafactor	S		S	0,7734	0,7081	5	3401	862
65	Adam	S	S		0,7838	0,7061	12	7191	768
66	AdamW	S	S		0,8015	0,7141	7	3520	572
67	Adadelta	S	S		0,8002	0,7121	4	2210	603
68	Adafactor	S	S		0,7813	**0,7210**	7	4260	768
69	Adam	S	S	S	0,8082	0,7021	11	6600	691
70	AdamW	S	S	S	0,8155	0,7010	5	2702	629
71	Adadelta	S	S	S	0,7998	0,7058	6	3095	609
72	Adafactor	S	S	S	0,7875	0,7134	10	6359	821
							Total	105426	15216

Fonte: Elaboração própria.

experimentos sem ajuste dos pesos. Essa técnica permite que o modelo se adapte melhor aos dados específicos do conjunto de treinamento.

Quanto ao uso de técnicas de aumento de dados, os resultados indicam que o seu uso não apresenta uma tendência clara de melhoria ou piora na acurácia. Isso sugere que, para o modelo NASNetLarge, o aumento de dados pode não ser tão crucial para o desempenho.

É importante observar que experimentos com configurações mais complexas, como o uso de *Transfer Learning*, ajuste dos pesos e técnicas de aumento de dados, geralmente requerem um tempo de treinamento mais longo. Isso deve ser considerado ao implementar o modelo em um ambiente de produção, onde o tempo de treinamento e teste é um fator importante.

No geral, os resultados dos experimentos com o modelo NASNetLarge indicam que o uso de Transfer Learning, ajuste dos pesos e técnicas de aumento de dados podem contribuir para um desempenho ligeiramente melhor, resultando em acurácias um pouco mais altas. No entanto, a escolha dessas técnicas deve levar em consideração o tempo de processamento envolvido, uma vez que configurações mais complexas podem exigir um tempo de treinamento mais longo.

Dentre os experimentos realizados na família NASNetLarge, o melhor resultado obtido foi alcançado na configuração identificada como ID 68. Esses resultados indicam que essa configuração obteve um bom desempenho no modelo generativo de textos descritivos para imagens médicas. A *loss*, que mensura a discrepância entre as previsões do modelo e os rótulos reais, apresentou um valor relativamente baixo, indicando uma boa capacidade de ajuste do

modelo aos dados. A configuração alcançou esses resultados após 7 épocas de treinamento, com um tempo total de treinamento de 4260 segundos e tempo de teste de 768 segundos.

4.2.4 Família VGG19

Na Tabela 8, é apresentado os resultados de diferentes configurações do modelo utilizando a rede VGG19 como a rede convolucional (CNN). Na sequencia, os resultados serão analisados considerando as variáveis já definidas.

Tabela 8 – Resultados da Família VGG19

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
73	Adam		S		0,7734	0,7056	16	9480	310
74	AdamW		S		0,7534	0,7166	7	3384	158
75	Adadelta		S		0,7375	0,7252	11	5194	150
76	Adafactor		S		0,7231	0,7242	6	2831	150
77	Adam		S	S	0,8118	0,7126	12	5693	150
78	AdamW		S	S	0,7517	0,7258	16	7615	152
79	Adadelta		S	S	0,7382	0,7318	5	2336	150
80	Adafactor		S	S	0,7266	0,7273	6	2805	151
81	Adam	S			0,6648	0,7510	10	4685	152
82	AdamW	S			0,6530	0,7485	4	1887	158
83	Adadelta	S			0,6530	0,7483	8	3879	156
84	Adafactor	S			0,6485	0,7503	5	2459	166
85	Adam	S		S	0,6276	**0,7628**	18	8930	163
86	AdamW	S		S	0,6509	0,7456	4	2014	172
87	Adadelta	S		S	0,6325	0,7525	10	5040	171
88	Adafactor	S		S	0,6158	0,7607	14	7091	176
89	Adam	S	S		0,6659	0,7558	11	5467	168
90	AdamW	S	S		0,6608	0,7580	7	3410	240
91	Adadelta	S	S		0,6787	0,7538	4	3495	153
92	Adafactor	S	S		0,6784	0,7477	7	3179	163
93	Adam	S	S	S	0,6608	0,7586	10	4369	150
94	AdamW	S	S	S	0,6380	0,7552	13	6118	170
95	Adadelta	S	S	S	0,6307	0,7564	10	6210	174
96	Adafactor	S	S	S	0,6224	0,7593	5	2512	177
							Total	110084	4080

Fonte: Elaboração própria.

Em relação aos otimizadores, observou-se que as configurações com Adam e AdamW geralmente apresentam valores de perda e acurácia mais favoráveis em comparação com as configurações com Adadelta e Adafactor.

Verificou-se que, em geral, a utilização do *Transfer Learning* e o ajuste dos pesos durante a etapa de treinamento melhoraram o desempenho do modelo, conforme evidenciado pelas configurações com ‘S’ nessas colunas.

O uso de técnicas de aumento de dados (IA) também mostra impacto nos resultados. As configurações com aumento de dados tendem a apresentar valores de perda mais baixos e acurácia mais alta em comparação com as configurações sem aumento de dados.

Quanto ao tempo de treinamento e teste, há variações significativas entre as diferentes configurações. O tempo de treinamento pode variar de pouco mais de 1.800 segundos a mais

de 9.000 segundos, enquanto o tempo de teste varia de cerca de 150 segundos a mais de 300 segundos.

Em resumo, a análise da Tabela 8 indica que a utilização da VGG19 em conjunto com o otimizador Adam e o uso de técnicas de aumento de dados pode levar a um bom desempenho na tarefa de geração de textos descritivos para imagens médicas. O *Transfer Learning* e o ajuste dos pesos também contribuem para melhorar o desempenho do modelo.

O melhor resultado obtido na Tabela 8 é representado pela configuração com ID 85. Nessa configuração, foi utilizada a arquitetura VGG19 em conjunto com o otimizador Adam, além da combinação de *Transfer Learning* e aumento de dados.

Essa configuração específica alcançou uma perda de 0,6276 e uma acurácia de 0,7628. Esses valores indicam um desempenho muito favorável na tarefa de geração de textos descritivos em imagens médicas.

A aplicação de Transfer Learning possibilitou aproveitar o conhecimento prévio aprendido pela rede VGG19 em uma tarefa relacionada. Isso pode levar a um melhor desempenho, uma vez que a rede já foi treinada em uma grande quantidade de dados de imagens gerais.

Além disso, o uso de técnicas de aumento de dados contribui para aumentar a quantidade e a diversidade dos dados de treinamento. Isso pode melhorar a capacidade do modelo de generalizar e lidar com variações nas imagens médicas, resultando em um desempenho mais robusto.

O otimizador Adam é conhecido por ser eficiente e eficaz no treinamento de redes neurais, adaptando a taxa de aprendizado de forma adaptativa para cada parâmetro do modelo. Essa característica pode ter contribuído para o desempenho superior alcançado nessa configuração.

Em resumo, o melhor resultado obtido na Tabela 8, representado pela configuração com ID 85, demonstra a importância de combinar diferentes estratégias, como *Transfer Learning*, aumento de dados e otimizadores eficientes para alcançar um desempenho notável na geração de textos descritivos em imagens médicas.

4.2.5 Família Xception

Na Tabela 9, é apresentado os resultados dos experimentos realizados com o modelo Xception em combinação com diferentes otimizadores de gradiente, *Transfer Learning*, aumento de dados e ajuste dos pesos durante o treinamento.

Tabela 9 – Resultados da Família Xception

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
97	Adam		S		0,7686	0,7215	16	2994	120
98	AdamW		S		0,7551	0,7223	9	1724	121
99	Adadelta		S		0,7416	0,7334	9	1728	118
100	Adafactor		S		0,7381	0,7289	7	1358	120
101	Adam		S	S	0,7961	0,7069	16	3174	121
102	AdamW		S	S	0,7709	0,7104	6	1244	134
103	Adadelta		S	S	0,7614	0,7154	6	1200	124
104	Adafactor		S	S	0,7687	0,7131	4	799	121
105	Adam	S			0,7521	0,7235	12	2342	124
106	AdamW	S			0,7493	0,7298	4	821	134
107	Adadelta	S			0,7416	0,7310	4	804	131
108	Adafactor	S			0,7381	0,7313	6	1208	132
109	Adam	S		S	0,7825	0,7306	20	4983	171
110	AdamW	S		S	0,7013	0,7449	4	870	139
111	Adadelta	S		S	0,6895	**0,7499**	6	1238	133
112	Adafactor	S		S	0,6848	0,7479	4	826	132
113	Adam	S	S		0,7112	0,7400	12	2371	133
114	AdamW	S	S		0,7237	0,7362	4	1003	124
115	Adadelta	S	S		0,7209	0,7354	4	1232	278
116	Adafactor	S	S		0,7134	0,7366	4	1483	274
117	Adam	S	S	S	0,6928	0,7359	20	3613	125
118	AdamW	S	S	S	0,6896	0,7413	4	772	141
119	Adadelta	S	S	S	0,6958	0,7393	6	1186	133
120	Adafactor	S	S	S	0,6816	0,7463	8	1599	142
							Total	40571	3426

Fonte: Elaboração própria.

Quando se trata de otimizadores de gradiente, é perceptível que diferentes otimizadores levam a resultados variados. Por exemplo, na configuração com o otimizador Adam, obteve-se uma perda de 0,7686 e uma acurácia de 0,7215, enquanto na configuração com o otimizador Adadelta, teve-se uma perda de 0,7416 e uma acurácia de 0,7334. Essa variação sugere que a escolha do otimizador pode ter um impacto significativo no desempenho do modelo.

Em geral, as configurações com *Transfer Learning* mostraram resultados ligeiramente melhores em termos de perda e acurácia em comparação com as configurações sem *Transfer Learning*. Por exemplo, a configuração com TL usando o otimizador Adam apresentou uma perda de 0,7961 e uma acurácia de 0,7069, enquanto a configuração correspondente sem TL obteve uma perda de 0,7521 e uma acurácia de 0,7235. Isso sugere que o uso de recursos pré-treinados, como os extratores de atributos do conjunto de dados *ImageNet*, pode melhorar a capacidade do modelo em capturar informações relevantes das imagens médicas.

Outro fator importante é o uso de técnicas de aumento de dados. Comparando as configurações com e sem aumento de dados, pode-se observar que, em alguns casos, o aumento de dados resulta em uma melhoria no desempenho do modelo. Por exemplo, a configuração com o otimizador Adafactor e o uso de aumento de dados (IA) apresenta uma perda de 0,7381 e uma acurácia de 0,7289, enquanto a configuração correspondente sem aumento de dados mostra uma perda de 0,7687 e uma acurácia de 0,7131. Isso sugere que o aumento de dados pode ajudar o modelo a aprender padrões mais robustos e a generalizar melhor.

O tempo de treinamento e teste registrado na Tabela 9 fornece informações sobre o tempo de processamento associado a cada configuração. É possível observar que o tempo varia consideravelmente entre as configurações, o que indica a importância de considerar o equilíbrio entre o desempenho do modelo e os recursos computacionais necessários.

No geral, a análise desses resultados ressalta a importância de explorar diferentes combinações de variáveis, como extratores de atributos, otimizadores de gradiente, uso de *Transfer Learning* e técnicas de aumento de dados, para obter uma compreensão mais profunda do desempenho e das limitações do modelo generativo de textos descritivos para imagens médicas.

O melhor modelo dessa família é o de ID 111 que foi configurado com a arquitetura Xception, o otimizador Adadelta, o uso de *Transfer Learning* e técnicas de aumento de dados.

Os resultados obtidos pelo modelo com ID 111 mostram um bom desempenho geral. Ele obteve uma perda relativamente baixa (0,6895) e uma acurácia consideravelmente alta (0,7499), indicando que o modelo foi capaz de aprender padrões relevantes nas imagens médicas e fazer previsões precisas. O tempo necessário para treinar o modelo foi de 1238 segundos. Esse valor indica o tempo de processamento associado ao treinamento do modelo, sendo relevante considerar os recursos computacionais disponíveis.

Com base nos resultados apresentados, este modelo demonstra potencial para gerar descrições textuais precisas e de alta qualidade para imagens médicas.

4.2.6 Família InceptionV3

Na Tabela 10, é apresentado os resultados de diferentes configurações utilizadas no experimento para o modelo InceptionV3. Observando os resultados, pode-se fazer as seguintes observações e discussões.

Os otimizadores AdamW, Adadelta e Adafactor mostraram acurácias próximas, com resultados ligeiramente superiores ao Adam e, em termos de tempo de treinamento e teste, os otimizadores AdamW e Adafactor apresentaram tempos menores em comparação com Adam e Adadelta.

A utilização de *Transfer Learning* e aumento de dados parece ter um efeito positivo na acurácia geral dos modelos. Os resultados mostram que a combinação de TL e IA levou a acurácias mais altas do que quando esses recursos foram usados separadamente. No entanto, é importante notar que o aumento de dados pode levar a um aumento no tempo de treinamento.

Tabela 10 – Resultados da Família InceptionV3

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
121	Adam		S		0,8028	0,7109	20	2202	129
122	AdamW		S		0,7700	0,7125	4	472	131
123	Adadelta		S		0,7584	0,7145	7	825	132
124	Adafactor		S		0,7457	0,7319	10	1170	131
125	Adam		S	S	0,8109	0,6992	17	2066	133
126	AdamW		S	S	0,8084	0,7054	4	487	132
127	Adadelta		S	S	0,7991	0,7070	9	1199	182
128	Adafactor		S	S	0,7926	0,6997	8	1056	167
129	Adam	S			0,7521	0,7283	12	1490	164
130	AdamW	S			0,7507	0,7300	4	500	169
131	Adadelta	S			0,7498	0,7358	6	749	165
132	Adafactor	S			0,7371	0,7315	5	635	179
133	Adam	S		S	0,7435	0,7213	13	1724	189
134	AdamW	S		S	0,7246	0,7315	12	1599	174
135	Adadelta	S		S	0,7044	0,7425	7	934	180
136	Adafactor	S		S	0,7118	**0,7472**	8	1059	187
137	Adam	S	S		0,7869	0,7324	12	1565	192
138	AdamW	S	S		0,7718	0,7217	4	528	182
139	Adadelta	S	S		0,7571	0,7285	8	1075	198
140	Adafactor	S	S		0,7541	0,7276	7	1478	216
141	Adam	S	S	S	0,7626	0,7210	12	2631	229
142	AdamW	S	S	S	0,7178	0,7337	12	2519	227
143	Adadelta	S	S	S	0,7238	0,7267	4	739	210
144	Adafactor	S	S	S	0,7320	0,7249	4	858	222
							Total	29562	4219

Fonte: Elaboração própria.

Isso indica que o uso de modelos pré-treinados e técnicas de aumento de dados é benéfico para melhorar o desempenho do modelo de geração de textos descritivos para imagens médicas.

O melhor modelo dessa família, identificado pelo ID 136, obteve uma perda de 0,7118 e uma acurácia de 0,7472, o que indica que o modelo foi capaz de gerar descrições textuais precisas para as imagens médicas utilizadas no experimento.

A configuração do melhor modelo incluiu o uso do *Transfer Learning* com a arquitetura InceptionV3, que é um modelo pré-treinado com o conjunto de dados *ImageNet*. Essa abordagem é eficaz para aproveitar os conhecimentos prévios adquiridos pelo modelo com um grande conjunto de imagens e aplicá-los a um novo domínio, como as imagens médicas. Isso permite que o modelo capture informações relevantes das imagens médicas e as utilize para gerar descrições textuais mais precisas. Além disso, o modelo também se beneficiou das técnicas de aumento de dados, o que ajudou a melhorar a capacidade do modelo de generalizar para diferentes variações nas imagens médicas.

Em termos de desempenho computacional, o modelo foi treinado por 8 épocas e o tempo de treinamento foi de 1059 segundos (aproximadamente 17 minutos). Já o tempo de teste foi de 187 segundos (cerca de 3 minutos). Esses tempos são importantes para avaliar a eficiência do modelo em termos de tempo de processamento.

Ao considerar todos esses fatores, pode-se concluir que o modelo com o ID 136 apresentou um bom desempenho na geração de textos descritivos para imagens médicas. A

combinação do *Transfer Learning* com a arquitetura InceptionV3, o otimizador Adafactor e as técnicas de aumento de dados permitiram que o modelo capturasse informações relevantes das imagens, gerando descrições textuais precisas com uma acurácia de 74,72%.

4.2.7 Família InceptionResNetV2

Ao analisar os resultados da Tabela 11, pode-se observar que o desempenho do modelo InceptionResNetV2 de acordo com diferentes combinações de otimizadores de gradiente, uso de *Transfer Learning*, técnicas de aumento de dados e ajuste dos pesos durante o treinamento. No entanto, não foi identificado um padrão claro de desempenho em relação a essas variáveis.

Tabela 11 – Resultados da Família InceptionResNetV2

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
145	Adam		S		0,8118	0,7066	12	3164	269
146	AdamW		S		0,8171	0,6901	7	1887	267
147	Adadelta		S		0,8048	0,6975	7	1890	272
148	Adafactor		S		0,7941	0,7117	5	1369	283
149	Adam		S	S	0,8011	0,7165	15	4679	264
150	AdamW		S	S	0,8145	0,6922	4	1149	307
151	Adadelta		S	S	0,8067	0,6940	6	1744	343
152	Adafactor		S	S	0,7871	0,7076	8	3213	456
153	Adam	S			0,8422	0,6967	12	4939	455
154	AdamW	S			0,8247	0,6920	5	1838	418
155	Adadelta	S			0,8167	0,7019	11	4628	472
156	Adafactor	S			0,8269	0,6915	4	2113	547
157	Adam	S		S	0,8184	0,6886	16	8287	489
158	AdamW	S		S	0,7907	0,6997	11	4793	504
159	Adadelta	S		S	0,7863	0,7073	6	3332	1245
160	Adafactor	S		S	0,7823	0,7054	7	3881	1211
161	Adam	S	S		0,8214	0,6962	16	4131	255
162	AdamW	S	S		0,8192	0,7011	9	2375	271
163	Adadelta	S	S		0,8087	0,7122	4	1047	262
164	Adafactor	S	S		0,7982	0,7057	6	1576	260
165	Adam	S	S	S	0,8305	0,6823	17	4530	265
166	AdamW	S	S	S	0,8246	0,6848	4	1095	313
167	Adadelta	S	S	S	0,8146	0,6906	9	2500	320
168	Adafactor	S	S	S	0,8136	0,6923	4	1133	330
							Total	71296	10077

Fonte: Elaboração própria.

Em relação aos otimizadores de gradiente, não houve diferenças significativas na acurácia e perda do modelo. Contudo, os otimizadores Adam e AdamW apresentaram resultados similares em termos de acurácia e perda, com desempenho consistente em várias combinações de variáveis. Já os otimizadores Adadelta e Adafactor obtiveram acurácia ligeiramente inferior em algumas configurações.

Em geral, o uso do *Transfer Learning* não apresentou um impacto significativo no desempenho do modelo e as configurações com e sem alcançaram resultados semelhantes em termos de acurácia e perda.

Não houve uma diferença significativa nos resultados em relação ao ajuste dos pesos

durante o treinamento dos modelos. Tanto as configurações com quanto sem o ajuste dos pesos durante o treinamento apresentaram desempenho similar em termos de acurácia e perda.

A inclusão de técnicas de aumento de dados mostrou um pequeno aumento na acurácia em alguns casos, mas não de forma consistente. Em algumas configurações, a aplicação de aumento de dados resultou em um aumento na acurácia, enquanto em outras não houve uma melhoria significativa. Em geral, o impacto do aumento de dados foi moderado.

É importante considerar que o tempo de treinamento e teste variou bastante para cada configuração. Como essa família possuiu resultados bem semelhantes, esse fato pode ser relevante ao escolher o otimizador de gradiente adequado e demais técnicas.

Portanto, como conclusão dessa análise, a família InceptionResNetV2 obteve resultados consistentes em termos de acurácia e perda em comparação aos demais modelos testados e não houve um modelo claramente melhor ou pior em relação as variáveis analisadas.

4.2.8 Família MobileNetV2

Na Tabela 12, é apresentado os resultados de diferentes configurações utilizadas no experimento para o modelo MobileNetV2. Observando os resultados, pode-se fazer as seguintes observações e discussões.

Tabela 12 – Resultados da Família MobileNetV2

ID	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
169	Adam		S		0,9017	0,6775	16	1209	76
170	AdamW		S		0,9028	0,6704	7	552	79
171	Adadelta		S		0,8984	0,6734	4	327	79
172	Adafactor		S		0,9001	0,6773	7	555	77
173	Adam		S	S	0,9104	0,6603	10	840	74
174	AdamW		S	S	0,9119	0,6839	7	592	81
175	Adadelta		S	S	0,9063	0,6779	9	833	84
176	Adafactor		S	S	0,8998	0,6809	5	467	83
177	Adam	S			0,8022	0,7108	11	935	85
178	AdamW	S			0,7965	0,7183	4	343	82
179	Adadelta	S			0,7940	0,7173	6	517	85
180	Adafactor	S			0,7759	0,7229	7	596	84
181	Adam	S		S	0,7684	0,7125	14	1304	85
182	AdamW	S		S	0,7910	0,7196	4	384	88
183	Adadelta	S		S	0,7795	0,7186	4	384	90
184	Adafactor	S		S	0,7715	0,7203	4	383	89
185	Adam	S	S		0,7996	0,7118	11	1016	104
186	AdamW	S	S		0,7855	0,7225	4	428	111
187	Adadelta	S	S		0,7622	0,7306	4	431	103
188	Adafactor	S	S		0,7712	0,7238	7	639	89
189	Adam	S	S	S	0,7532	0,7150	14	1350	90
190	AdamW	S	S	S	0,7434	0,7282	7	794	112
191	Adadelta	S	S	S	0,7261	**0,7345**	4	400	93
192	Adafactor	S	S	S	0,7353	0,7253	10	1166	113
							Total	16445	2134

Fonte: Elaboração própria.

Observou-se que o otimizador de gradiente AdamW tende a apresentar melhores

resultados em termos de perda e acurácia, especialmente quando o *Transfer Learning* e o ajuste de pesos durante o treinamento são aplicados.

O uso do *Transfer Learning* foi explorado, utilizando-se extratores de atributos pré-treinados com o conjunto de dados *ImageNet*. Os experimentos indicam que o uso do *Transfer Learning* tende a melhorar ligeiramente a acurácia do modelo, especialmente quando combinado com o ajuste de pesos durante o treinamento. Isso sugere que o conhecimento prévio adquirido com o *ImageNet* pode ser transferido com sucesso para a tarefa de geração de textos descritivos para imagens médicas e que adaptar os pesos dos extratores de atributos durante o treinamento pode ajudar o modelo a aprender características específicas das imagens médicas.

Analisando a influência do aumento de dados, percebeu-se que os resultados mostram que, em termos de perda, os experimentos com aumento de dados geralmente apresentaram valores ligeiramente maiores em comparação com os experimentos sem aumento de dados. Isso sugere que o aumento de dados pode introduzir alguma variabilidade nos resultados e aumentar a complexidade do modelo. No entanto, é importante destacar que a diferença na perda não é significativa.

Quanto à influência do aumento de dados na acurácia, os resultados mostram que, em alguns casos, o uso de aumento de dados levou a um aumento na acurácia em comparação com os experimentos sem aumento de dados. No entanto, em outras configurações, não houve uma melhoria significativa na acurácia com o uso de aumento de dados.

É importante notar que, embora o uso de aumento de dados possa aumentar a acurácia em alguns casos, também pode aumentar o tempo necessário para treinar o modelo. Os resultados indicam que os experimentos com aumento de dados geralmente requerem mais tempo de treinamento em comparação com os experimentos sem aumento de dados.

Em resumo, as análises em relação ao aumento de dados indicam que essa técnica pode ter um impacto positivo na acurácia do modelo em determinadas configurações. No entanto, é necessário considerar o *trade-off* entre o aumento na acurácia e o aumento no tempo de treinamento.

O melhor modelo da família MobileNetV2, identificado pelo ID 191 na Tabela 12, corresponde à configuração com o otimizador Adadelta, utilização de *Transfer Learning* e o uso de aumento de dados.

A análise dos resultados desso melhor modelo mostra que ele alcançou uma perda de 0,7261 e uma acurácia de 0,7345. Esses valores indicam um bom desempenho do modelo na

tarefa de geração dos textos descritivos de imagens médicas.

Além disso, é interessante observar que esse modelo foi treinado em 4 épocas, o que indica uma convergência relativamente rápida durante o processo de treinamento. Isso pode ser atribuído à combinação eficaz do otimizador Adadelta com as outras variáveis do modelo.

Em relação ao tempo de treinamento e teste, o modelo levou cerca de 400 segundos (ou aproximadamente 6 minutos e 40 segundos) para o treinamento e 93 segundos (ou cerca de 1 minuto e 33 segundos) para o teste. Esses tempos são considerados razoáveis e indicam uma eficiência computacional adequada para o modelo.

Em conclusão, o melhor modelo (ID 191) obteve bons resultados em termos de perda, acurácia e tempo de treinamento. Ele representa uma configuração promissora para a tarefa de geração de textos descritivos de imagens médicas.

4.2.9 Melhores Resultados de Cada Família

Na Tabela 13, é apresentado um resumo do experimento com o melhor resultado de cada família de modelos utilizados para extrair os atributos das imagens médicas, exceto para a família InceptionResNetV2 que não teve um melhor resultado claramente definido. Essa tabela permite uma comparação direta entre as diferentes famílias de modelos e seus respectivos desempenhos. Portanto, pode-se observar a relevância do uso ou não de diferentes abordagens para a ocorrência do melhor resultado.

Tabela 13 – Melhores Resultados de Cada Família

ID	CNN	OPTZ	TL	TR	IA	Loss	Acc	Epochs	Training Time (s)	Test Time (s)
20	DenseNet201	Adafactor	S	S		0,7190	0,7610	6	2205	420
36	ResNet152V2	Adafactor	S			0,7413	0,7359	9	3771	452
68	NASNetLarge	Adafactor	S	S		0,7813	0,7210	7	4260	768
85	VGG19	Adam	S		S	0,6276	**0,7628**	18	8930	163
111	Xception	Adadelta	S		S	0,6895	0,7499	6	1238	133
136	InceptionV3	Adafactor	S		S	0,7118	0,7472	8	1059	187
191	MobileNetV2	Adadelta	S	S	S	0,7261	0,7345	4	400	93

Fonte: Elaboração própria.

A utilização do otimizador Adafactor é encontrada em 4 dos 7 experimentos com melhores resultados (IDs 20, 36, 68 e 136), enquanto o otimizador Adadelta é usado em dois experimentos (IDs 111 e 191). Já o otimizador Adam é usado em apenas um experimento (ID 85). Esses resultados sugerem que utilizar o otimizador Adafactor pode estar relacionada a melhores desempenhos, uma vez que aparece em 4 dos melhores resultados. No entanto, é importante notar que outros fatores, como a arquitetura do extrator de atributos, o uso de *Transfer Learaning*, o ajuste dos pesos durante o treinamento e o uso de aumento de dados também podem influenciar

os resultados.

Todos os melhores resultados utilizaram *Transfer Learning* com uma rede pré-treinada em um conjunto de dados do *ImageNet*, indicando a importância dessa técnica para o desempenho geral dos modelos. Isso significa que as redes foram inicializadas com pesos pré-treinados, o que contribuiu para melhores resultados devido ao aprendizado prévio em tarefas de extração de atributos de imagens.

Em relação ao ajuste dos pesos durante o treinamento, dos 7 melhores resultados listados, 3 deles (IDs 20, 68 e 191) indicam que houve o uso dessa técnica. Essa informação sugere que o ajuste dos pesos pode não ter sido uma estratégia relevante para melhorar o desempenho dos modelos. Isso indica que os pesos inicializados via *Transfer Learning* pré-treinados do *ImageNet* já tem uma boa capacidade de extrair atributos das imagens médicas.

Quanto a utilização de aumento de dados, pode-se observar que a técnica esteve presente em 4 dos 7 experimentos (IDs 85, 111, 136 e 191). Isso sugere que o aumento de dados pode ter desempenhado um papel importante na melhoria do desempenho desses modelos, pois pode melhorar a capacidade do modelo de generalizar e lidar com variações nas imagens médicas, resultando em um desempenho mais robusto.

Por fim, os tempos de treinamento e teste variam significativamente entre os diferentes modelos. É importante observar que esses tempos podem ser influenciados por vários fatores, como a complexidade da rede, o tipo de otimizador de gradiente e a utilização ou não de técnicas de aumento de dados. Os tempos de treinamento variam de 400 segundos (ID 191 - MobileNetV2) a 8930 segundos (ID 85 - VGG19). Já os tempos de teste variam de 93 segundos (ID 191 - MobileNetV2) a 768 segundos (ID 68 - NASNetLarge). É interessante observar que o tempo de treinamento e teste nem sempre está diretamente relacionado ao melhor desempenho alcançado pelos modelos.

O melhor resultado obtido entre todas as famílias, levando em conta a melhor acurácia, corresponde a variante de ID 85. Essa variante consiste em um modelo VGG19 que foi treinado utilizando *Transfer Learning* e aplicação de aumento de dados. O valor de perda alcançado foi de 0,6276 e a acurácia obtida foi de 0,7628.

Com base nos dados disponíveis na Tabela 13, podemos observar que o modelo VGG19 obteve uma acurácia bastante alta, indicando que ele foi capaz de gerar textos descritivos de imagens médicas na maioria das amostras de teste.

O uso de *Transfer Learning*, combinado com a aplicação de aumento de dados, pode

ter contribuído para o bom desempenho do modelo VGG19. O *Transfer Learning* permite que o modelo se beneficie do conhecimento prévio aprendido em tarefas de extração de atributos de imagem. Além disso, a aplicação de aumento de dados pode ter ajudado a aumentar a variabilidade dos dados de treinamento, tornando o modelo mais robusto e generalizável.

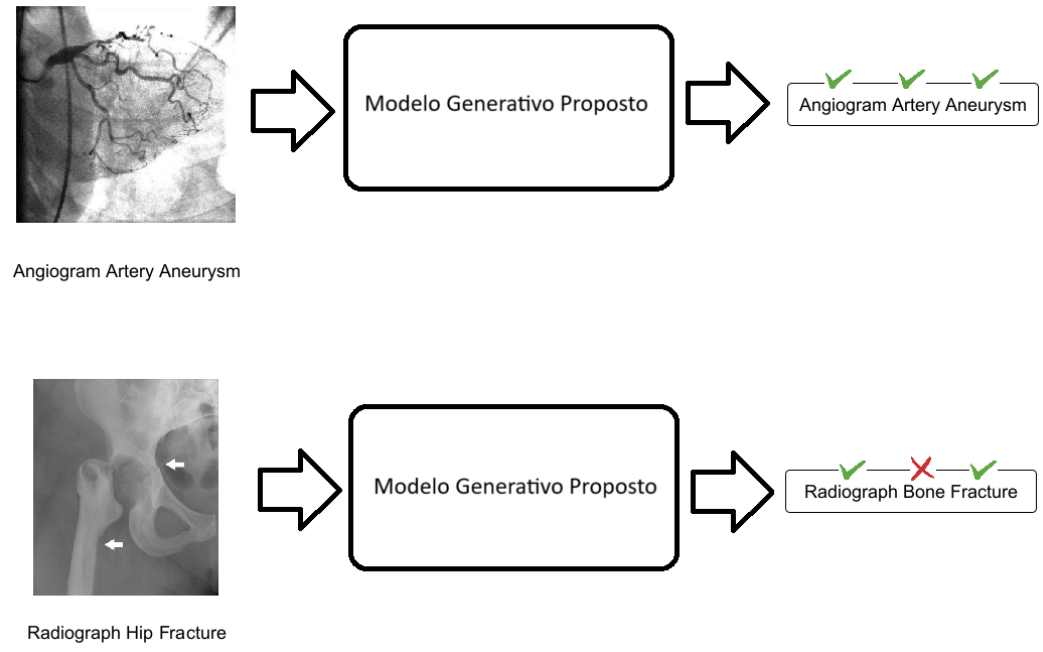
No entanto, é importante ressaltar que para uma análise completa do desempenho do melhor modelo, é necessário métricas adicionais como precisão, taxa de acerto e medida F1 para que se possa realizar análises estatísticas para determinar a significância dos resultados. Este é o objetivo da próxima seção.

4.3 Análises do Melhor Modelo

Para o melhor modelo obtido no experimento, foi realizada uma avaliação detalhada de seu desempenho através da construção de matrizes de confusão para cada parte dos textos gerados: Tipo de exame, parte do corpo e problema identificado.

A título de exemplo, tem-se na Figura 6 uma representação de duas imagens quaisquer do conjunto de dados de teste e os respectivos textos descritivos verdadeiros e gerados pelo melhor modelo, o qual alcançou uma acurácia de 0,7628 e uma pontuação BLEU-1 de 0,5387.

Figura 6 – Exemplos de Imagens de Entrada e Descrições Verdadeiras e Geradas



Fonte: Elaboração própria.

Essas matrizes permitiram visualizar as escolhas corretas e incorretas feitas pelo

melhor modelo na construção de cada parte dos textos descritivos das imagens médicas. Com base nessas matrizes, indicadores-chave foram calculados, fornecendo medidas objetivas da qualidade das descrições textuais geradas pelo modelo, servindo para avaliar seu desempenho. Portanto, cada matriz de confusão possui uma tabela associada que sintetizam esses indicadores.

A matriz de confusão é uma ferramenta valiosa para avaliar o desempenho de um modelo de classificação, apresentando as previsões do modelo em relação às classes reais. Nas matrizes de confusão deste trabalho, as classes reais estão representadas no eixo vertical, enquanto as classes previstas estão representadas no eixo horizontal.

4.3.1 Matriz de Confusão do Tipo de Exame

A matriz de confusão representada pela Tabela 14 apresenta as previsões do melhor modelo em relação ao tipo de exame das imagens. Observando essa matriz de confusão, pode-se tirar algumas conclusões.

Tabela 14 – Matriz de Confusão do Tipo de Exame

Real/Predição	'ct'	'radiograph'	'scan'	'tomography'	'xray'	'angiogram'	'angiography'	'abdominal'	contrast'
'ct'	108	2	10	3	0	0	0	0	0
'radiograph'	1	32	0	0	12	0	0	0	0
'scan'	19	0	26	5	0	0	0	0	1
'tomography'	16	0	9	0	0	0	0	0	0
'xray'	0	0	0	0	11	0	0	0	0
'angiogram'	1	0	0	0	0	9	1	0	0
'angiography'	3	0	0	0	0	7	0	0	0
'abdominal'	4	0	1	0	0	0	0	0	0
'contrast'	0	0	0	0	0	3	0	0	0

Fonte: Elaboração própria.

4.3.1.1 Análise Qualitativa

O modelo apresentou um desempenho bastante satisfatório para a classe 'ct' (tomografia computadorizada), com 108 exemplos corretamente classificados. Houve apenas algumas confusões, com 2 exemplos sendo erroneamente classificados como 'radiograph' (radiografia), 10 como 'scan' (varredura) e 3 como 'tomography'.

A classe 'radiograph' também obteve um bom desempenho, com 32 exemplos corretamente classificados. No entanto, houve algumas confusões, especialmente com a classe 'xray' (raios X), onde 12 exemplos de radiografias foram erroneamente classificados. Entretanto, raios X e radiografia podem ser considerados termos intercambiáveis nesse contexto.

O modelo apresentou um desempenho não tão bom para a classe 'scan', com 26

exemplos corretamente classificados. No entanto, ocorreram algumas confusões, com 19 exemplos sendo classificados erroneamente como ‘ct’ e 5 exemplos como ‘tomography’ (tomografia).

A classe ‘tomography’ obteve um desempenho péssimo, com nenhum exemplo sendo classificado corretamente. Houve confusão, onde 9 exemplos foram confundidos com a classe ‘scan’ e 16 exemplos sendo erroneamente classificados como ‘ct’. Entretanto, tomografia e tomografia computadorizada podem ser considerados termos intercambiáveis nesse contexto.

Entretanto, a classe ‘xray’ obteve um desempenho excelente, com 11 exemplos corretamente classificados. Não houve confusões com outras classes nessa categoria.

Para as classes ‘angiogram’ (angiograma) e ‘angiography’ (angiografia), ambas tiveram resultados satisfatórios, com a maioria dos exemplos corretamente classificados. No entanto, 7 exemplos foram erroneamente classificados como ‘ct’.

A classe ‘abdominal’ teve um desempenho péssimo, com nenhum exemplo sendo classificado corretamente. Houve confusão, onde 4 exemplos foram confundidos com a classe ‘ct’ e 1 exemplo sendo erroneamente classificado como ‘scan’.

A classe ‘contrast’ também teve um desempenho ruim. Todos os exemplos foram erroneamente classificados como ‘angiogram’, sem ocorrência de confusões com outras classes.

4.3.1.2 Análise Quantitativa

Na Tabela 15, é apresentada as métricas da matriz de confusão do melhor modelo em relação ao tipo de exame das imagens. Analisando essa tabela, é possível avaliar o desempenho do melhor modelo e tirar algumas conclusões.

Tabela 15 – Métricas da Matriz de Confusão do Tipo de Exame

Classe	P	N	TP	FP	TN	FN	TPR	TNR	PPV	NPV	ACC	F1
‘ct’	123	161	108	44	117	15	0,88	0,73	0,71	0,89	0,79	0,79
‘radiograph’	45	239	32	2	237	13	0,71	0,99	0,94	0,95	0,95	0,81
‘scan’	51	233	26	20	213	25	0,51	0,91	0,57	0,89	0,84	0,54
‘tomography’	25	259	0	8	251	25	0,00	0,97	0,00	0,91	0,88	-
‘xray’	11	273	11	12	261	0	1,00	0,96	0,48	1,00	0,96	0,65
‘angiogram’	11	273	9	10	263	2	0,82	0,96	0,47	0,99	0,96	0,60
‘angiography’	10	274	0	1	273	10	0,00	1,00	0,00	0,96	0,96	-
‘abdominal’	5	279	0	0	279	5	0,00	1,00	-	0,98	0,98	-
‘contrast’	3	281	0	1	280	3	0,00	1,00	0,00	0,99	0,99	-

Fonte: Elaboração própria.

Ao analisar as métricas da matriz de confusão para todas as classes fornecidas, é possível observar uma variação considerável nos resultados. Vamos discutir cada classe em um contexto geral, analisando se os valores são considerados bons ou ruins.

Quanto a classe ‘ct’, tem-se um desempenho relativamente bom. A sensibilidade

e a especificidade estão acima de 0,7, o que significa que o modelo é capaz de identificar corretamente a maioria dos casos positivos e negativos. A precisão também está em torno de 0,71, o que indica que a maioria dos casos classificados como positivos é realmente positiva.

Em relação a classe 'radiograph', o desempenho é bastante positivo. A sensibilidade e a especificidade estão acima de 0,7 e 0,9, respectivamente, indicando um bom desempenho em identificar corretamente casos positivos e negativos. A precisão é alta, em torno de 0,94, o que sugere que a maioria das classificações positivas é correta. Além disso, o F1 Score de 0,81 indica um bom equilíbrio entre precisão e sensibilidade.

Os valores para o termo 'scan' são um pouco mais baixos em comparação com os termos anteriores. A sensibilidade e a especificidade estão acima de 0,5, indicando um desempenho razoável na identificação de casos positivos e negativos. No entanto, a precisão e o F1 Score estão em torno de 0,57, sugerindo que há espaço para melhoria na classificação correta dos casos positivos.

Os valores para o termo 'tomography' são baixos. A sensibilidade e a precisão são 0, o que significa que o modelo não identificou corretamente nenhum caso positivo. O F1 Score é indefinido, pois o modelo não identificou nenhuma vez essa classe corretamente, o que indica um desempenho muito baixo para esse termo.

Os valores para o termo 'xray' mostram um desempenho misto. A sensibilidade é alta, indicando que o modelo identificou corretamente todos os casos positivos. A especificidade também é alta, mostrando que o modelo é capaz de identificar corretamente a maioria dos casos negativos. No entanto, a precisão é relativamente baixa, sugerindo que muitos dos casos classificados como positivos podem ser falsos positivos. O F1 Score indica um equilíbrio moderado entre precisão e sensibilidade.

Para o termo 'angiogram', os valores mostram um desempenho razoável. A sensibilidade indica que o modelo conseguiu identificar corretamente a maioria dos casos positivos. A especificidade é alta, indicando que a maioria dos casos negativos foi identificada corretamente. No entanto, a precisão é relativamente baixa, o que sugere que algumas das classificações positivas podem ser falsos positivos. O F1 Score indica um equilíbrio moderado entre precisão e sensibilidade.

Para o termo 'angiography', a sensibilidade e a precisão são 0, o que indica que o modelo não conseguiu identificar corretamente nenhum caso positivo. O F1 Score é indefinido, pois o modelo não identificou nenhuma vez essa classe corretamente, o que indica um

desempenho muito baixo para esse termo.

Para o termo ‘abdominal’, a sensibilidade e a precisão são 0, o que indica que o modelo não conseguiu identificar corretamente nenhum caso positivo. O F1 Score é indefinido, pois o modelo não identificou nenhuma vez essa classe corretamente, o que indica um desempenho muito baixo para esse termo.

Para o termo ‘contrast’, a sensibilidade é 0, o que indica que o modelo não conseguiu identificar corretamente nenhum caso positivo. O F1 Score é indefinido, pois o modelo não identificou nenhuma vez essa classe corretamente, o que indica um desempenho muito baixo para esse termo.

4.3.1.3 Discussão dos Resultados

Em geral, o modelo mostrou um desempenho promissor na classificação dos tipos de exames médicos. No entanto, algumas classes apresentaram maior dificuldade de serem corretamente classificadas, resultando em confusões com outras classes. Esses resultados fornecem *insights* valiosos sobre os pontos fortes e as limitações do modelo generativo de textos descritivos para imagens médicas proposto no estudo.

Considerando a intercambialidade de alguns termos (‘tomography’ equivale a ‘ct, por exemplo), pode-se concluir que os resultados obtidos são consistentes e indicam um desempenho geralmente bom do modelo na tarefa de classificação dos tipos de exames médicos.

As métricas de acurácia, sensibilidade, especificidade, precisão e valor preditivo negativo apresentaram valores relativamente altos, indicando que o modelo foi capaz de realizar classificações corretas tanto para os exemplos positivos quanto para os exemplos negativos na maioria dos casos analisados. Isso é um indicativo positivo da capacidade do modelo em aprender padrões relevantes nas imagens médicas e gerar descrições textuais precisas.

É importante observar que valores de sensibilidade iguais a 0 em alguns termos indica que o modelo não conseguiu identificar nenhum caso positivo para esses termos. Além disso, a presença de valores indefinidos para a precisão e o F1 Score indica limitações na interpretação dessas métricas devido à falta de previsões positivas. Em geral, a análise das métricas mostra um desempenho variado para os diferentes termos. Alguns termos apresentam valores positivos de sensibilidade, especificidade, precisão e F1 Score, indicando um bom desempenho na identificação correta dos casos positivos e negativos. No entanto, outros termos têm desempenho inferior, com valores baixos ou indefinidos para algumas métricas.

No entanto, é necessário ter em mente que existem variações nos resultados, o que indica que o desempenho do modelo pode ser influenciado por diferentes fatores, como a distribuição dos dados ou as características específicas de cada classe. Essas variações podem ser observadas em algumas métricas, como a sensibilidade, especificidade e precisão.

Portanto, embora os resultados sejam bons de forma geral, é importante continuar refinando e otimizando o modelo, explorando diferentes abordagens e técnicas, como ajuste fino dos pesos, aumento de dados e seleção de atributos relevantes. Além disso, é fundamental realizar validações adicionais e testes em conjuntos de dados externos para verificar a generalização do modelo.

4.3.2 Matriz de Confusão da Parte do Corpo

A matriz de confusão representada pela Tabela 16 apresenta as previsões do melhor modelo em relação a parte do corpo das imagens. Observando essa matriz de confusão, pode-se tirar algumas conclusões.

Tabela 16 – Matriz de Confusão da Parte do Corpo

Real/Predito	'chest'	'abdoman'	'artery'	'lung'	'lobe'	'head'	'liver'	'pelvi'	'bone'	'pulmonary'	'aortic'	'vein'	'hip'	'neck'	'femoral'
'chest'	39	3	0	10	1	0	0	0	1	0	0	0	0	0	0
'abdoman'	0	47	2	2	0	0	0	0	1	0	0	0	0	0	0
'artery'	1	10	24	0	0	0	0	0	0	0	0	0	0	0	0
'lung'	12	6	2	16	0	0	0	0	0	0	0	0	0	0	0
'lobe'	0	22	1	1	1	0	0	0	0	0	0	0	0	0	0
'head'	0	14	0	0	0	1	0	0	0	0	0	0	0	0	0
'liver'	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
'pelvi'	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
'bone'	2	1	0	0	0	0	0	2	6	0	0	0	0	0	0
'pulmonary'	2	0	1	4	0	0	0	0	0	0	0	0	0	0	0
'aortic'	0	3	1	0	1	0	0	0	0	0	0	0	0	0	0
'vein'	0	9	0	0	0	1	0	0	1	0	0	0	0	0	0
'hip'	0	0	0	0	0	0	0	2	1	0	0	0	1	0	0
'neck'	1	0	0	0	0	0	0	6	3	0	0	0	0	0	0
'femoral'	0	0	0	0	0	0	0	1	5	0	0	0	0	0	1

Fonte: Elaboração própria.

4.3.2.1 Análise Qualitativa

Na classificação de 'chest', o modelo apresenta um desempenho geralmente bom, com 39 casos corretamente previstos. Existem algumas confusões com 'abdoman' (3 casos incorretos) e 'lung' (10 casos incorretos). No entanto, a maioria das previsões para 'chest' é correta, indicando um bom desempenho na identificação dessa parte do corpo.

O modelo tem um desempenho satisfatório na classificação de 'abdoman', com 47 casos corretamente previstos. Há algumas confusões com 'artery' (2 casos incorretos) e 'lung' (2 casos incorretos). No geral, o modelo é capaz de identificar adequadamente 'abdoman', mas

há algumas instâncias em que ocorrem confusões com outras partes do corpo.

O modelo mostra um desempenho relativamente bom na classificação de ‘artery’, com 24 casos corretamente previstos. No entanto, observa-se algumas confusões com ‘abdoman’, onde foram identificados incorretamente 10 casos, e com ‘chest’, onde ocorreu uma única previsão incorreta. Essas confusões podem ser atribuídas à proximidade anatômica entre as artérias e essas regiões específicas do corpo, o que pode dificultar a distinção visual entre elas. Portanto, é importante continuar refinando o modelo para melhorar a precisão nessas distinções e reduzir as confusões entre ‘artery’, ‘abdoman’ e ‘chest’.

Na classificação de ‘lung’, com 16 casos corretamente previstos, o modelo apresentou um desempenho razoável. No entanto, ocorreram algumas confusões com ‘chest’ (12 casos incorretos) e ‘abdoman’ (6 casos incorretos). Essas confusões podem ser atribuídas ao fato de que os pulmões estão localizados tanto na região do tórax quanto do abdômen. Essa sobreposição anatômica pode tornar visualmente desafiador para o modelo distinguir corretamente entre ‘lung’, ‘chest’ e ‘abdoman’. Portanto, é necessário aprimorar a capacidade do modelo de diferenciar com precisão entre essas partes do corpo, considerando características mais específicas das imagens médicas relacionadas aos pulmões.

O modelo enfrentou dificuldades na classificação de ‘lobe’, com apenas 1 caso corretamente previsto. A maioria das previsões para ‘lobe’ foi confundida com ‘abdoman’ (22 casos incorretos). Essas confusões podem ser atribuídas ao fato de que ‘lobe’, no conjunto de dados utilizado, está relacionado aos lóbulos direito e esquerdo do fígado, que estão localizados no abdômen. Essa proximidade anatômica entre ‘lobe’ e ‘abdoman’ pode tornar desafiador para o modelo distinguir corretamente entre eles, especialmente considerando as características visuais das imagens médicas. Portanto, é necessário aprimorar a capacidade do modelo em diferenciar com precisão os lóbulos do fígado (‘lobe’) do abdômen (‘abdoman’) por meio de características mais específicas e discriminativas presentes nas imagens médicas.

O desempenho do modelo na classificação de ‘head’ mostra que há espaço para melhorias, já que apenas 1 caso foi corretamente previsto. Houve uma confusão significativa com ‘abdoman’ (14 casos incorretos). Essas confusões podem ser justificada pelo fato de que, no contexto do conjunto de dados utilizado, ‘head’ se refere a uma parte específica do pâncreas que está localizada no abdômen. Essa associação entre ‘head’ e o pâncreas, que é uma estrutura abdominal, pode levar a confusões na classificação, uma vez que o modelo pode interpretar visualmente as características presentes nas imagens de forma semelhante para ambas as classes.

Dessa forma, a proximidade anatômica e a sobreposição de características visuais entre os termos podem dificultar a distinção precisa entre ‘head’ e ‘abdoman’ para o modelo. Portanto, é necessário considerar abordagens que explorem características mais específicas relacionadas ao pâncreas, além de técnicas de processamento de imagens mais avançadas, a fim de melhorar a capacidade do modelo em diferenciar corretamente essas partes do corpo.

O modelo obteve um desempenho insatisfatório na classificação de ‘liver’, não tendo previsto corretamente nenhum caso dessa classe e todos os casos foram erroneamente classificadas como ‘abdoman’. As confusões observadas na classificação de ‘liver’ pode ser atribuída ao fato de que o fígado está localizado no abdômen. Essa proximidade anatômica entre o fígado e outras estruturas abdominais pode levar a confusões na classificação, uma vez que as características visuais dessas estruturas podem se sobrepor nas imagens médicas. O fígado é um órgão grande e complexo que ocupa uma posição central no abdômen. Sua forma, textura e localização podem variar entre os indivíduos e também em diferentes imagens médicas. Essa variação pode dificultar a segmentação e identificação precisa do fígado, levando a confusões com outras partes do corpo presentes na mesma região anatômica.

A classe ‘pelvi’ apresentou resultados de classificação consistentes, com todas as previsões corretas feitas pelo modelo. Isso indica um desempenho satisfatório na identificação de imagens que correspondem à região pélvica do corpo. O fato de o modelo ter classificado corretamente todas as imagens associadas à classe ‘pelvi’ indica sua capacidade de reconhecer as características visuais relevantes dessa região específica do corpo. Isso sugere que as características distintivas das imagens médicas da pelve foram bem aprendidas e capturadas pelo modelo durante o treinamento.

A classe ‘bone’ apresentou um desempenho moderado na classificação das imagens médicas. O modelo foi capaz de identificar corretamente 6 casos relacionados a essa classe, porém com confusões em outras categorias, como ‘abdoman’ e ‘chest’. A confusão observada entre ‘bone’ e outras classes pode ser explicada pela presença de características visuais semelhantes ou sobreposição nas imagens. Por exemplo, nas imagens médicas do abdômen (‘abdoman’) e do tórax (‘chest’), os ossos podem estar parcialmente visíveis, resultando em confusões na classificação.

A classe ‘pulmonary’ apresentou um desempenho ruim na classificação das imagens médicas. O modelo foi incapaz de identificar corretamente 4 casos relacionados a essa classe, mas se confundiu com outras categorias, como ‘chest’ e ‘lung’. Como a classe ‘pulmonary’ está

associada às estruturas pulmonares, como os pulmões, brônquios e vasos sanguíneos pulmonares, as confusões observadas com outras classes podem ser atribuídas à similaridade visual entre as estruturas pulmonares e outras estruturas adjacentes no tórax, como o próprio ‘chest’ e o ‘lung’. Essa sobreposição visual pode dificultar a distinção precisa entre as categorias durante a classificação.

A classe ‘aortic’ apresentou um desempenho ruim na classificação das imagens médicas. O modelo não foi capaz de identificar corretamente nenhum caso relacionado a essa classe, apresentando confusões com outras categorias, como ‘abdoman’ e ‘artery’. A classe ‘aortic’ está associada à aorta, que é a maior artéria do corpo humano e desempenha um papel crucial no transporte de sangue para diversas partes do organismo. As confusões observadas com outras classes podem ser explicadas pela similaridade visual entre a aorta e outras estruturas, como as artérias em geral (‘artery’) e a região abdominal (‘abdoman’). Essa sobreposição visual e características compartilhadas podem dificultar a distinção precisa entre as categorias durante a classificação.

Em se tratando da classe ‘vein’, o modelo mostrou um desempenho ruim, com nenhum caso corretamente previsto. No entanto, houve algumas confusões com outras classes, em especial como a classe ‘abdoman’. As confusões podem ser atribuídas ao fato de que os dados de veias do conjunto de dados estão relacionadas ao fígado, que está localizado na região abdominal. Essa proximidade anatômica e visual entre as veias hepáticas e outras estruturas abdominais pode dificultar a distinção precisa durante o processo de classificação.

A classe ‘hip’ representa a região do quadril nas imagens médicas. O modelo apresentou um desempenho não-satisfatório na classificação do quadril, com 1 caso corretamente previsto. No entanto, houve algumas confusões com outras classes, como ‘pelvi’ e ‘bone’. Essas confusões podem ser justificadas pelo fato de que o quadril e a pelve estão intimamente relacionados anatomicamente, e a distinção precisa entre essas estruturas pode ser desafiadora. Além disso, o quadril é composto por várias estruturas ósseas, como o osso do quadril (íleo, ísquio e púbis) e o fêmur, que podem se sobrepor a outras regiões ósseas próximas.

A classe ‘neck’ representa a região do pescoço nas imagens médicas. O modelo obteve resultados ruins na classificação dessa classe, com nenhum caso corretamente previsto, ocorrendo confusões com outras classes, como ‘pelvi’ e ‘bone’ e ‘chest’. A confusão com ‘chest’ pode ser atribuída a transição entre o pescoço e o abdômen pode não ser claramente distinta, especialmente quando se trata de imagens com sobreposição de estruturas. Já a confusão com

‘pelvi’ e ‘bone’ pode ser devido à presença de ossos cervicais no pescoço, como as vértebras cervicais. Essas estruturas ósseas podem ser semelhantes às encontradas em outras partes do corpo, dificultando a classificação precisa do pescoço.

A classe ‘femoral’ representa a região da artéria femoral nas imagens médicas. Embora o modelo tenha tido alguns acertos na classificação dessa classe, houve confusões significativas com as classes ‘bone’ e ‘pelvi’. A confusão com a classe ‘bone’ pode ser atribuída ao fato do fêmur ser um osso, o que dificulta a distinção precisa entre as duas classes. Já a confusão com a classe ‘pelvi’ pode ser explicada pela localização próxima da artéria femoral à região pélvica. Em algumas imagens, a artéria femoral pode estar parcialmente dentro da região pélvica, o que pode levar a confusões entre as duas classes.

4.3.2.2 Análise Quantitativa

Na Tabela 17, é apresentada as métricas da matriz de confusão do melhor modelo em relação a parte do corpo das imagens. Analisando essa tabela, é possível avaliar o desempenho do melhor modelo e tirar algumas conclusões.

Tabela 17 – Métricas da Matriz de Confusão da Parte do Corpo

Classe	P	N	TP	FP	TN	FN	TPR	TNR	PPV	NPV	ACC	F1
‘chest’	54	230	39	18	212	15	0,72	0,92	0,68	0,93	0,88	0,70
‘abdoman’	52	232	47	78	154	5	0,90	0,66	0,38	0,97	0,71	0,53
‘artery’	35	249	24	7	242	11	0,69	0,97	0,77	0,96	0,94	0,73
‘lung’	36	248	16	17	187	20	0,44	0,92	0,48	0,90	0,85	0,46
‘lobe’	25	259	1	2	257	24	0,04	0,99	0,33	0,91	0,91	0,07
‘head’	15	269	1	1	268	14	0,07	1,00	0,50	0,95	0,95	0,12
‘liver’	10	274	0	0	274	10	0,00	1,00	-	0,96	0,96	-
‘pelvi’	2	282	2	11	271	0	1,00	0,96	0,15	1,00	0,96	0,27
‘bone’	11	273	6	12	261	5	0,55	0,96	0,33	0,98	0,94	0,41
‘pulmonary’	7	277	0	0	277	7	0,00	1,00	-	0,98	0,98	-
‘aortic’	5	279	0	0	279	5	0,00	1,00	-	0,98	0,98	-
‘vein’	11	273	0	0	273	11	0,00	1,00	-	0,96	0,96	-
‘hip’	4	280	1	0	280	3	0,25	1,00	1,00	0,99	0,99	0,40
‘neck’	10	274	0	0	274	10	0,00	1,00	-	0,96	0,96	-
‘femoral’	7	277	1	0	277	6	0,14	1,00	1,00	0,98	0,98	0,25

Fonte: Elaboração própria.

Ao analisar as métricas da matriz de confusão para todas as classes fornecidas, é possível observar uma variação considerável nos resultados. Vamos discutir cada classe em um contexto geral, analisando se os valores são considerados bons ou ruins.

As métricas apresentadas para a classe ‘chest’ indicam um desempenho relativamente bom. O True Positive Rate (TPR) de 0,72 e o True Negative Rate (TNR) de 0,92 mostram uma taxa satisfatória de identificação correta tanto para casos positivos quanto negativos. O Positive Predictive Value (PPV) de 0,68 indica que cerca de 68% dos casos classificados como positivos

são realmente positivos. O Accuracy (ACC) de 0,88 também é bastante razoável. O F1 score de 0,70, que é uma média harmônica entre precisão e recall, indica um equilíbrio adequado entre essas métricas.

Os resultados para a classe ‘abdoman’ não são tão favoráveis. Embora o TPR de 0,90 seja alto, sugerindo uma boa taxa de identificação correta de casos positivos, o TNR de 0,66 indica uma taxa relativamente baixa de identificação correta de casos negativos. O PPV de 0,38 indica que apenas 38% dos casos classificados como positivos são realmente positivos. A Accuracy (ACC) de 0,71 também é relativamente baixa. O F1 score de 0,53 mostra um desempenho moderado, mas ainda assim há espaço para melhorias.

As métricas para a classe ‘artery’ são geralmente positivas. O TPR de 0,69 indica uma taxa razoável de identificação correta de casos positivos, enquanto o TNR de 0,97 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,77 é relativamente alto, indicando que 77% dos casos classificados como positivos são realmente positivos. A Accuracy (ACC) de 0,94 também é bastante satisfatória. O F1 score de 0,73 mostra um bom equilíbrio entre precisão e recall.

Para a classe ‘lung’, os resultados não são tão positivos. O TPR de 0,44 indica uma taxa relativamente baixa de identificação correta de casos positivos, enquanto o TNR de 0,92 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,48 indica que apenas 48% dos casos classificados como positivos são realmente positivos. A Accuracy (ACC) de 0,85 também é moderada. O F1 score de 0,46 mostra um desempenho abaixo do ideal em termos de equilíbrio entre precisão e recall.

As métricas para a classe ‘lobe’ revelam um desempenho bastante baixo. O TPR de 0,04 indica uma taxa extremamente baixa de identificação correta de casos positivos. Por outro lado, o TNR de 0,99 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,33 indica que apenas 33% dos casos classificados como positivos são realmente positivos. A Accuracy (ACC) de 0,91 é moderada, mas o F1 score de 0,07 indica um desempenho muito baixo em termos de equilíbrio entre precisão e recall.

As métricas para a classe ‘head’ também não apresentam um desempenho satisfatório. O TPR de 0,07 indica uma taxa muito baixa de identificação correta de casos positivos, enquanto o TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,50 indica que apenas 50% dos casos classificados como positivos são realmente positivos. A Accuracy (ACC) de 0,95 é razoável, mas o F1 score de 0,12 indica um desempenho baixo em

termos de equilíbrio entre precisão e recall.

Para a classe ‘liver’, não há casos positivos (TP = 0), o que impede o cálculo do PPV e do F1 score. No entanto, o TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos, e o Negative Predictive Value (NPV) de 0,96 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,96 também é considerada boa.

As métricas para a classe ‘pelvi’ revelam um desempenho misto. O TPR de 1,00 indica uma taxa alta de identificação correta de casos positivos, enquanto o TNR de 0,96 sugere uma alta taxa de identificação correta de casos negativos. No entanto, o PPV de 0,15 indica que apenas 15% dos casos classificados como positivos são realmente positivos. O NPV de 1,00 indica que todos os casos negativos foram corretamente identificados. A Accuracy (ACC) de 0,96 é razoável, e o F1 score de 0,27 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

As métricas para a classe ‘bone’ indicam um desempenho moderado. O TPR de 0,55 sugere uma taxa razoável de identificação correta de casos positivos, enquanto o TNR de 0,96 indica uma alta taxa de identificação correta de casos negativos. O PPV de 0,33 indica que apenas 33% dos casos classificados como positivos são realmente positivos. O NPV de 0,98 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,94 é considerada boa. O F1 score de 0,41 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

Para a classe ‘pulmonary’, não há casos positivos (TP = 0), o que impede o cálculo do PPV e do F1 score. No entanto, o TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos. O NPV de 0,98 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,98 é considerada boa.

Assim como na classe ‘pulmonary’, não há casos positivos (TP = 0) para a classe ‘aortic’, impossibilitando o cálculo do PPV e do F1 score. O TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos. O NPV de 0,98 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,98 é considerada boa.

Da mesma forma que nas classes anteriores, não há casos positivos (TP = 0) para a classe ‘vein’, impossibilitando o cálculo do PPV e do F1 score. O TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos. O NPV de 0,96 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,96 é considerada razoável.

As métricas para a classe ‘hip’ indicam um desempenho relativamente bom. O TPR

de 0,25 indica uma taxa moderada de identificação correta de casos positivos, enquanto o TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 1,00 indica que todos os casos classificados como positivos são realmente positivos. O NPV de 0,99 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,99 é considerada boa. O F1 score de 0,40 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

Para a classe 'neck', não há casos positivos (TP = 0), o que impede o cálculo do PPV e do F1 score. O TNR de 1,00 sugere uma alta taxa de identificação correta de casos negativos. O NPV de 0,96 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,96 é considerada razoável.

As métricas para a classe 'femoral' indicam um desempenho moderado. O TPR de 0,14 sugere uma taxa relativamente baixa de identificação correta de casos positivos, enquanto o TNR de 1,00 indica uma alta taxa de identificação correta de casos negativos. O PPV de 1,00 indica que todos os casos classificados como positivos são realmente positivos. O NPV de 0,98 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,98 é considerada boa. O F1 score de 0,25 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

4.3.2.3 Discussão dos Resultados

Em resumo, a análise da matriz de confusão da parte do corpo revelou um desempenho mediano do modelo na classificação das partes do corpo em imagens médicas. Embora tenham ocorrido algumas confusões entre as classes, é importante destacar que o modelo foi capaz de identificar corretamente o contexto geral em muitos casos, embora nem sempre tenha utilizado as palavras exatas para descrever as partes do corpo.

O modelo apresentou um desempenho relativamente bom na classificação de algumas classes, como 'chest', 'abdoman' e 'artery', o que indica sua capacidade de identificar estruturas vasculares nas imagens médicas. No entanto, foram observadas confusões com outras classes, como 'lobe', 'head' e 'liver', o que pode ser atribuído à proximidade anatômica ou à falta de distinção visual clara entre essas estruturas.

Apesar das confusões, é encorajador que o modelo tenha sido capaz de capturar o contexto geral e fornecer informações relevantes sobre as partes do corpo presentes nas imagens médicas. Isso sugere que o modelo tem potencial para auxiliar em tarefas de descrição e

classificação, mesmo que precise ser refinado para melhorar a precisão na escolha das palavras específicas.

No entanto, é importante ressaltar que o desempenho do modelo pode ser aprimorado por meio de abordagens mais avançadas, como a utilização de técnicas de processamento de imagens mais sofisticadas, a incorporação de informações contextuais adicionais e o treinamento em conjuntos de dados mais diversificados e balanceados.

Portanto, embora o modelo tenha apresentado um desempenho mediano na classificação das partes do corpo em imagens médicas, há espaço para melhorias e refinamentos que podem permitir uma descrição mais precisa e específica das estruturas anatômicas presentes nas imagens.

Em resumo, a análise das métricas da matriz de confusão revelou resultados variados para as diferentes classes. Algumas classes apresentaram um desempenho satisfatório, com altas taxas de identificação correta de casos positivos e negativos, além de valores positivos para as métricas de precisão, recall e F1 score.

No entanto, outras classes apresentaram um desempenho inferior, com baixas taxas de identificação correta de casos positivos, valores baixos para as métricas de precisão e recall, e F1 scores baixos.

Além disso, algumas classes não tiveram casos positivos na amostra, o que dificulta a avaliação adequada de métricas como precisão e F1 score para esses casos. No entanto, essas classes mostraram uma alta taxa de identificação correta de casos negativos e altos valores de Negative Predictive Value (NPV), indicando que os casos negativos foram classificados corretamente.

4.3.3 Matriz de Confusão do Problema Identificado

A matriz de confusão representada pela Tabela 18 apresenta as previsões do melhor modelo em relação ao problema identificado das imagens. Observando essa matriz de confusão, pode-se tirar algumas conclusões.

4.3.3.1 Análise Qualitativa

Na classe ‘nodule’, o modelo teve um desempenho relativamente bom na classificação, com 30 casos corretamente identificados. Embora o modelo tenha acertado a maioria das predições, houve várias confusões com outras classes, como ‘hepatic’, ‘aneurysm’, ‘tumor’ e

Tabela 18 – Matriz de Confusão do Problema Identificado

Real/Predito	'nodule'	'hepatic'	'aneurysm'	'fracture'	'normal'	'tumor'	'pancreatic'	'effusion'	'cystic'	'hematoma'	'cyst'
'nodule'	30	3	2	0	0	4	0	4	8	0	0
'hepatic'	0	5	1	1	0	0	16	0	22	0	0
'aneurysm'	1	1	15	0	1	0	3	0	8	1	0
'fracture'	0	0	0	30	3	0	0	0	1	0	0
'normal'	0	0	9	1	21	0	0	0	1	0	0
'tumor'	1	1	0	1	0	9	2	0	3	1	0
'pancreatic'	0	2	0	0	0	0	15	0	2	3	0
'effusion'	10	0	1	0	0	5	0	0	4	0	0
'cystic'	0	0	2	0	0	0	6	2	9	0	0
'hematoma'	0	1	0	0	0	0	3	0	3	0	0
'cyst'	0	0	0	0	0	0	2	0	4	0	0

Fonte: Elaboração própria.

'cystic'. Essas confusões podem ser justificadas pela natureza complexa dos nódulos, que podem variar em tamanho, forma e localização. No entanto, é importante destacar que mesmo com as confusões, o modelo foi capaz de realizar várias predições corretas para casos de nódulos. Isso indica que o modelo conseguiu capturar certas características distintivas dos nódulos, mesmo que tenha havido dificuldades em casos mais complexos.

A classe 'hepatic' apresentou um desempenho ruim em termos de classificação. O modelo teve algumas predições corretas, mas também houve um número significativo de confusões com outras classes, como 'cystic' e 'pancreatic'. Essa confusão pode ser explicada pela sobreposição visual do fígado e pâncreas nas imagens médicas, levando a previsões incorretas devido a uma sobreposição de características dificultando a diferenciação precisa pelo modelo.

A classe 'aneurysm' apresentou um desempenho razoável em termos de classificação, com 15 predições corretas. No entanto, também houve algumas confusões com outras classes, como 'pancreatic' e 'cystic'. As confusões podem ser justificadas pelo fato de certos tipos de cistos podem exibir características semelhantes a aneurismas, tornando a distinção entre essas classes desafiadora. Apesar das confusões, o modelo conseguiu identificar corretamente uma parte significativa dos casos de aneurismas.

Já na classe 'fracture', o modelo apresentou um bom desempenho na classificação dessa classe, com 30 casos corretamente identificados e poucas confusão com outras classes.

O mesmo aconteceu para a classe 'normal', com 21 casos corretamente identificados e poucas confusão com outras classes.

Entretanto, a classe 'tumor' foi uma das classes com desempenho razoável em termos de classificação. O modelo conseguiu fazer um número significativo de predições corretas para tumores. No entanto, também houve algumas confusões com outras classes, como 'cystic' e 'pancreatic'. As confusões podem ser atribuídas ao fato de certos tipos de tumores podem ter semelhanças visuais com lesões hepáticas e cistos, dificultando a distinção precisa entre essas classes. Apesar das confusões, o modelo foi capaz de identificar corretamente 9 casos de tumores,

o que indica sua capacidade de capturar certas características distintivas dessa condição.

Para a classe ‘pancreatic’, o modelo teve um desempenho relativamente bom na classificação dessa classe, com 15 casos corretamente identificados. No entanto, ocorreram algumas confusões com outras classes, como ‘hepatic’, ‘cystic’ e ‘hematoma’. Essa confusão pode ser explicada, em parte, pela sobreposição visual do fígado e pâncreas nas imagens médicas, levando a previsões incorretas devido a uma sobreposição de características dificultando a diferenciação precisa pelo modelo.

Na classe ‘effusion’, houve apenas confusões com outras classes, como ‘nodule’, ‘cystic’ e ‘tumor’. Essas confusões podem ser justificadas pela presença de características visuais sobrepostas ou semelhantes entre as condições. Por exemplo, algumas efusões podem apresentar características de massa ou lesão que se assemelham a nódulos ou cistos. Da mesma forma, algumas efusões podem ser visualmente semelhantes a tumores em termos de aparência nas imagens médicas.

A classe ‘cystic’ apresentou um desempenho mediano em termos de classificação, com 9 predições corretas. No entanto, algumas confusões ocorreram com outras classes, como ‘pancreatic’. Essa confusão pode ser justificadas pela presença de características visuais compartilhadas ou sobrepostas entre as condições. Por exemplo, alguns cistos podem apresentar características de nódulos ou lesões hepáticas, tornando a distinção entre essas classes desafiadora. Da mesma forma, a presença de cistos em órgãos como o pâncreas pode ser visualmente semelhante a outras condições, como tumores pancreáticos.

A classe ‘hematoma’ apresentou um desempenho ruim, com todas as predições erradas. Houve uma confusão com ‘cystic’ que pode ser justificada pelo fato de que alguns hematomas podem apresentar aparência semelhante a cistos ou nódulos, especialmente quando há acúmulo de fluido ou sangue em uma região específica. Da mesma forma, a presença de hematomas em órgãos hepáticos pode se assemelhar a lesões hepáticas, o que justificaria a confusão com a classe ‘pancreatic’, tornando a diferenciação entre as classes desafiadora.

A classe ‘cyst’ apresentou um desempenho geralmente bom em termos de classificação, com a maioria das predições corretas, considerando que os termos ‘cyst’ e ‘cystic’ são intercambiáveis. No entanto, alguma confusão ocorreu com a classe ‘pancreatic’. Essas confusões podem ser justificadas pelo fato que alguns cistos podem compartilhar características lesões hepáticas, dificultando a distinção entre essas classes. Apesar da confusão, o modelo foi capaz de identificar corretamente a maioria dos casos de cistos, indicando sua capacidade de

capturar características distintivas dessa condição.

4.3.3.2 Análise Quantitativa

Na Tabela 19, são apresentas as métricas da matriz de confusão do melhor modelo em relação ao problema identificado das imagens. Analisando essa tabela, é possível avaliar o desempenho do melhor modelo e tirar algumas conclusões.

Tabela 19 – Métricas da Matriz de Confusão do Problema Identificado

Classe	P	N	TP	FP	TN	FN	TPR	TNR	PPV	NPV	ACC	F1
'nodule'	51	233	30	12	221	21	0,59	0,95	0,71	0,91	0,88	0,65
'hepatic'	45	239	5	8	231	40	0,11	0,97	0,38	0,85	0,83	0,17
'aneurysm'	30	254	15	15	239	15	0,50	0,94	0,50	0,94	0,89	0,50
'fracture'	34	250	30	3	247	4	0,88	0,99	0,91	0,98	0,98	0,90
'normal'	32	252	21	4	248	11	0,66	0,98	0,84	0,96	0,95	0,74
'tumor'	18	266	9	9	257	9	0,50	0,97	0,50	0,97	0,94	0,50
'pancreatic'	22	262	15	32	230	7	0,68	0,88	0,32	0,97	0,86	0,43
'effusion'	20	264	0	6	258	20	0,00	0,98	0,00	0,93	0,91	-
'cystic'	19	265	9	56	209	10	0,47	0,79	0,14	0,95	0,77	0,21
'hematoma'	7	277	0	5	272	7	0,00	0,98	0,00	0,97	0,96	-
'cyst'	6	278	0	0	278	6	0,00	1,00	-	0,98	0,98	-

Fonte: Elaboração própria.

Ao analisar as métricas da matriz de confusão para todas as classes fornecidas, é possível observar uma variação considerável nos resultados. Vamos discutir cada classe em um contexto geral, analisando se os valores são considerados bons ou ruins.

A classe 'nodule' apresenta um desempenho razoável. O TPR de 0,59 indica uma taxa moderada de identificação correta de casos positivos, enquanto o TNR de 0,95 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,71 indica que cerca de 71% dos casos classificados como positivos são realmente positivos. O NPV de 0,91 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,88 é considerada boa. O F1 score de 0,65 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

A classe 'hepatic' apresenta um desempenho inferior. O TPR de 0,11 indica uma taxa baixa de identificação correta de casos positivos, enquanto o TNR de 0,97 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,38 indica que apenas cerca de 38% dos casos classificados como positivos são realmente positivos. O NPV de 0,85 indica uma proporção razoável de casos negativos classificados corretamente. A Accuracy (ACC) de 0,83 é considerada razoável. O baixo F1 score de 0,17 indica um desempenho deficiente em termos de equilíbrio entre precisão e recall.

A classe 'aneurysm' mostra um desempenho razoável. O TPR de 0,50 indica uma

taxa moderada de identificação correta de casos positivos, enquanto o TNR de 0,94 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,50 indica que cerca de 50% dos casos classificados como positivos são realmente positivos. O NPV de 0,94 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,89 é considerada boa. O F1 score de 0,50 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

A classe 'fracture' apresenta um desempenho geralmente bom. O TPR de 0,88 indica uma alta taxa de identificação correta de casos positivos, enquanto o TNR de 0,99 sugere uma taxa muito alta de identificação correta de casos negativos. O PPV de 0,91 indica que cerca de 91% dos casos classificados como positivos são realmente positivos. O NPV de 0,98 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,98 é considerada boa. O F1 score de 0,90 mostra um desempenho sólido em termos de equilíbrio entre precisão e recall.

A classe 'normal' apresenta um desempenho razoável. O TPR de 0,66 indica uma taxa moderada de identificação correta de casos positivos, enquanto o TNR de 0,98 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,84 indica que cerca de 84% dos casos classificados como positivos são realmente positivos. O NPV de 0,96 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,95 é considerada boa. O F1 score de 0,74 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

A classe 'tumor' apresenta um desempenho razoável. O TPR de 0,50 indica uma taxa moderada de identificação correta de casos positivos, enquanto o TNR de 0,97 sugere uma alta taxa de identificação correta de casos negativos. O PPV de 0,50 indica que cerca de 50% dos casos classificados como positivos são realmente positivos. O NPV de 0,97 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,94 é considerada boa. O F1 score de 0,50 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

A classe 'pancreatic' apresenta um desempenho inferior. O TPR de 0,68 indica uma taxa moderada de identificação correta de casos positivos, enquanto o TNR de 0,88 sugere uma taxa relativamente baixa de identificação correta de casos negativos. O PPV de 0,32 indica que apenas cerca de 32% dos casos classificados como positivos são realmente positivos. O NPV de 0,97 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy

(ACC) de 0,86 é considerada razoável. O F1 score de 0,43 mostra um desempenho moderado em termos de equilíbrio entre precisão e recall.

A classe ‘effusion’ possui um número de casos positivos igual a zero, o que dificulta a avaliação adequada das métricas de precisão, recall e F1 score. O TNR de 0,98 indica uma alta taxa de identificação correta de casos negativos. O PPV de 0,00 indica que não há casos classificados como positivos. O NPV de 0,93 indica uma proporção alta de casos negativos classificados corretamente. A Accuracy (ACC) de 0,91 é considerada razoável.

A classe ‘cystic’ apresenta um desempenho inferior. O TPR de 0,47 indica uma taxa moderada de identificação correta de casos positivos, enquanto o TNR de 0,79 sugere uma taxa relativamente baixa de identificação correta de casos negativos. O PPV de 0,14 indica que apenas cerca de 14% dos casos classificados como positivos são realmente positivos. O NPV de 0,95 indica uma alta proporção de casos negativos classificados corretamente. A Accuracy (ACC) de 0,77 é considerada razoável. O baixo F1 score de 0,21 indica um desempenho deficiente em termos de equilíbrio entre precisão e recall.

A classe ‘hematoma’ possui um número de casos positivos igual a zero, o que dificulta a avaliação adequada das métricas de precisão, recall e F1 score. O TNR de 0,98 indica uma alta taxa de identificação correta de casos negativos. O PPV de 0,00 indica que não há casos classificados como positivos. O NPV de 0,97 indica uma proporção alta de casos negativos classificados corretamente. A Accuracy (ACC) de 0,96 é considerada boa.

A classe ‘cyst’ possui um número de casos positivos igual a zero, o que dificulta a avaliação adequada das métricas de precisão, recall e F1 score. O TNR de 1,00 indica uma taxa muito alta de identificação correta de casos negativos. O PPV de 0,00 indica que não há casos classificados como positivos. O NPV de 0,98 indica uma proporção alta de casos negativos classificados corretamente. A Accuracy (ACC) de 0,98 é considerada boa.

4.3.3.3 Discussão dos Resultados

O modelo apresentou resultados variados na classificação das diferentes classes de problemas identificados em imagens médicas. Para algumas classes, como ‘nodule’, ‘aneurysm’, ‘fracture’ e ‘normal’, o desempenho foi relativamente bom, com um número significativo de predições corretas. No entanto, houve confusões com outras classes, indicando a sobreposição de características visuais entre diferentes condições médicas.

Por outro lado, algumas classes, como ‘hepatic’, ‘effusion’ e ‘hematoma’, apresenta-

ram um desempenho mais desafiador. Houve um número considerável de confusões com outras classes, evidenciando a sobreposição de características visuais e a dificuldade em distinguir essas condições.

Quanto as métricas da matriz de confusão, algumas classes apresentam desempenho razoável, como 'nodule', 'aneurysm', 'fracture' e 'normal', com taxas moderadas de identificação correta de casos positivos, altas taxas de identificação correta de casos negativos e valores de F1 score moderados. Destaca-se a classe 'fracture', que apresenta um desempenho geralmente bom, com alta taxa de identificação correta de casos positivos, altas taxas de identificação correta de casos negativos e um F1 score sólido. Já as classes 'hematoma' e 'cyst' possuem um número de casos positivos igual a zero, o que dificulta a avaliação adequada de algumas métricas.

Por outro lado, as classes 'hepatic', 'effusion' e 'hematoma' apresentam desempenho inferior, com baixas taxas de identificação correta de casos positivos e/ou baixos valores de F1 score.

Para melhorar a classificação nessas classes, pode-se empregar abordagens mais avançadas, como técnicas de processamento de imagem específicas e um conjunto de dados mais abrangente que contemple exemplos mais variados.

4.4 Comparação do Melhor Modelo com a Literatura

Na Tabela 20, é apresentada uma comparação resumida de alguns modelos encontrados na literatura com este trabalho. Analisando essa tabela, juntamente com os outros resultados apresentados anteriormente, é possível tirar algumas conclusões.

Tabela 20 – Comparação do Melhor Modelo com a Literatura

Referência	BLEU-1	Conjunto de Dados	Entrada	Saída
(PARK *et al.*, 2020)	0,3293	IU X-RAY	Raios X	Descrições detalhadas de raios X do tórax
(HUANG *et al.*, 2021)	0,2190	DeepEyeNet	Imagens de retinas e palavras-chave	Descrições detalhadas de imagens de retinas
(PARK *et al.*, 2021)	0,3731	PEIR GROSS e ICLEF-CAPTION	Raios X	Descrições detalhadas de raios X do tórax
(SINGH *et al.*, 2022)	?	VQA-Med	Imagens Diversas	Uma única palavra entre 7 possíveis
(BEDDIAR *et al.*, 2022b)	?	ImageCLEFmed	Imagens diversas e palavras-chave	Descrições detalhadas de imagens diversas
(SONG *et al.*, 2022)	0,5230 e 0,5700	Open-i e LGK	Raios X e Ultrasons	Descrições simples, de 1 palavra
(LEE *et al.*, 2022)	0,5064	IU X-RAY	Raios X	Descrições detalhadas de raios X do tórax
(SINGH *et al.*, 2023)	0,5160	Open-i	Raios X	Descrições detalhadas de raios X do tórax
Este Trabalho	0,5387	ROCO	Imagens diversas do corpo	Descrições simples de diferentes modalidades de imagem e condições médicas

Fonte: Elaboração própria.

Uma lacuna na literatura relacionada à geração de descrições de imagens médicas é

a falta de estudos que explorem o desempenho de modelos específicos nesse contexto. Muitos estudos não investigam como diferentes arquiteturas de modelos de extração de atributos e técnicas de geração de descrições se comparam em termos de qualidade das descrições geradas. Isso limita a compreensão sobre quais modelos são mais adequados para a tarefa e quais técnicas podem levar a melhores resultados.

Nesse sentido, o presente trabalho contribui para a área ao avaliar e comparar o desempenho de diferentes modelos específicos em relação à qualidade das descrições geradas. Ao utilizar uma variedade de modelos de extração de atributos e a abordagem Transformador, este estudo proporciona uma análise mais abrangente e permite identificar quais modelos podem ser mais eficazes na geração de descrições em imagens médicas.

Outra lacuna é a falta de generalização dos modelos de geração de descrições para diferentes modalidades de imagem e condições médicas. Muitos estudos se concentram em uma única modalidade de imagem, como radiografias ou ressonâncias magnéticas, ou em condições médicas específicas, como descrições de raios X do tórax. Isso limita a aplicabilidade dos modelos propostos em cenários mais amplos e dificulta sua utilização em diferentes contextos clínicos.

No entanto, este trabalho aborda essa lacuna explorando a generalização dos modelos de geração de descrições para diferentes modalidades de imagem médica, como radiografias, ressonâncias magnéticas ou tomografias computadorizadas, e para uma variedade de condições médicas. Essa abordagem amplia a aplicabilidade dos modelos propostos, permitindo que eles sejam utilizados em diferentes cenários clínicos e oferecendo maior flexibilidade na geração de descrições em imagens médicas.

Contudo, um aspecto importante a ser considerado é o fato de que ter um modelo mais generalista diminui a complexidade das descrições geradas e leva a um aumento na confusão, especialmente quando se trata de descrever nuances específicas de certas modalidades de imagem ou condições médicas. Portanto, é crucial encontrar um equilíbrio entre a generalização e a capacidade de capturar características detalhadas e relevantes das imagens médicas com o objetivo de auxiliar médicos e profissionais de saúde em diagnósticos e tomadas de decisão.

5 CONCLUSÕES E TRABALHOS FUTUROS

A geração automática de descrições para imagens médicas tem se mostrado uma área de pesquisa promissora, com o potencial de auxiliar os profissionais de saúde na interpretação e análise de exames clínicos. A implementação de soluções desse tipo pode resultar em benefícios significativos para pacientes e profissionais, agilizando a tomada de decisão e aprimorando a qualidade do atendimento médico. Neste trabalho, explorou-se o desenvolvimento e avaliação de um modelo generativo de descrições de imagens médicas generalista utilizando o conjunto de dados ROCO.

Primeiramente, foi analisado o atual estado da arte, identificando abordagens baseadas em técnicas de processamento de linguagem natural e aprendizado de máquina, como CNN*s*, RNNs e Transformadores. Essas abordagens têm demonstrado resultados promissores na geração de descrições para imagens médicas, embora a maioria dos estudos tenha se concentrado em imagens de domínios específicos.

Entretanto, identificou-se algumas lacunas na literatura, como a falta de estudos que explorem o desempenho de modelos específicos para geração de descrições médicas, a necessidade de avaliação objetiva da qualidade das descrições geradas, a falta de generalização dos modelos para diferentes modalidades de imagem e condições médicas, bem como a falta de transparência e interpretabilidade dos modelos.

Para abordar alguns desses problemas, adotou-se uma estratégia metodológica que combinou técnicas de processamento de linguagem natural e modelos de reconhecimento de imagens para extrair atributos relevantes das imagens médicas, os quais foram alimentados em um modelo generativo baseado em redes neurais.

Para isso, explorou-se a aplicabilidade do MobileNetV2, DenseNet201, ResNet152V2, NASNetLarge, VGG19, Xception, InceptionV3 e InceptionResNetV2 para geração de descrições médicas buscando a generalização dos modelos para diferentes modalidades de imagem e condições médicas.

Além disso, explorou-se as técnicas *Transfer Learning* e *Image Augmentation* para analisar suas influência na performance do modelo generativo. A avaliação do modelo gerado foi feita por meio da acurácia e BLEU. Essa avaliação permitiu verificar a qualidade e a adequação das descrições geradas pelo modelo proposto.

Embora se tenha obtido resultados promissores na geração de descrições de imagens médicas, a qualidade das descrições geradas ainda pode ser limitada. As descrições podem

apresentar erros semânticos ou falta de detalhes específicos relevantes. Essa limitação pode ser justificada pela disponibilidade e representatividade dos dados do ROCO, que podem influenciar os resultados obtidos. Além disso, as técnicas utilizadas para a geração de descrições podem não capturar completamente a complexidade das imagens médicas e sua interpretação clínica. Essas limitações devem ser consideradas ao interpretar e generalizar os resultados deste estudo.

Como sugestão para pesquisas futuras, ressalta-se a importância de explorar ainda mais a influência de diferentes técnicas e abordagens, como os *Vision Transformers* (ViTs) e os *Large Language Models* (LLMs). Uma linha de pesquisa promissora envolve o uso de arquiteturas de redes neurais mais avançadas, como redes neurais adversariais generativas (GANs) e arquiteturas híbridas que combinem elementos de visão e linguagem de maneira mais eficaz.

Além disso, é fundamental avançar na interpretabilidade dos modelos gerativos, desenvolvendo métodos e ferramentas que permitam compreender e controlar o processo de geração de descrições. A interpretabilidade é essencial para garantir a confiabilidade e a utilidade das aplicações que fazem uso desses modelos em contextos críticos, como a medicina.

Outro ponto a ser explorado é o emprego de conjuntos de dados ainda mais amplos e diversificados, abrangendo uma variedade de domínios e tipos de dados. Isso ajudará a melhorar a capacidade dos modelos de generalização e adaptação a diferentes tarefas e cenários.

Adicionalmente, é sugerido que se busque uma métrica mais adequada para avaliar a qualidade das descrições geradas, levando em consideração critérios específicos de relevância, coesão, e coerência textual, especialmente quando se trata de aplicações em que a qualidade das descrições é crítica.

Por fim, uma direção de pesquisa importante seria a realização de uma validação clínica abrangente para aprofundar a análise comparativa entre as descrições geradas pelos modelos e as fornecidas por especialistas em áreas como a medicina. Isso permitirá avaliar não apenas a qualidade das descrições, mas também sua utilidade prática e segurança em contextos clínicos, contribuindo para a adoção segura e eficaz dessas tecnologias inovadoras.

Em suma, este estudo contribui para o avanço do conhecimento na área da geração automática de descrições de imagens médicas, fornecendo um modelo generativo eficaz e abrindo caminhos para futuras pesquisas nesse campo. Através da combinação de técnicas de processamento de linguagem natural e aprendizado de máquina, pôde-se contribuir um pouco para a área e promover a utilização de tecnologias inovadoras que beneficiem a prática clínica e a qualidade dos cuidados de saúde.

REFERÊNCIAS

AGGARWAL, A. K. Learning texture features from GLCM for classification of brain tumor MRI images using random forest classifier. **WSEAS TRANSACTIONS ON SIGNAL PROCESSING**, World Scientific and Engineering Academy and Society (WSEAS), v. 18, p. 60–63, abr. 2022. Disponível em: https://doi.org/10.37394/232014.2022.18.8.

AL-KHUZAIE, M. Y.; ZEARAH, S. A.; MOHAMMED, N. J. Developing an efficient VGG19-based model and transfer learning for detecting acute lymphoblastic leukemia (ALL). In: **2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)**. IEEE, 2023. Disponível em: https://doi.org/10.1109/hora58378.2023.10156679.

AYESHA, H.; IQBAL, S.; TARIQ, M.; ABRAR, M.; SANAULLAH, M.; ABBAS, I.; REHMAN, A.; NIAZI, M. F. K.; HUSSAIN, S. Automatic medical image interpretation: State of the art and future directions. **Pattern Recognition**, Elsevier BV, v. 114, p. 107856, jun. 2021. Disponível em: https://doi.org/10.1016/j.patcog.2021.107856.

BEDDIAR, D.-R.; OUSSALAH, M.; SEPPäNEN, T. Automatic captioning for medical imaging (MIC): a rapid review of literature. **Artificial Intelligence Review**, Springer Science and Business Media LLC, v. 56, n. 5, p. 4019–4076, set. 2022. Disponível em: https://doi.org/10.1007/s10462-022-10270-w.

BEDDIAR, D. R.; OUSSALAH, M.; SEPPäNEN, T.; JENNANE, R. ACapMed: Automatic captioning for medical imaging. **Applied Sciences**, MDPI AG, v. 12, n. 21, p. 11092, nov. 2022. Disponível em: https://doi.org/10.3390/app122111092.

BURGER, W.; BURGE, M. J. **Digital Image Processing**. Springer London, 2016. Disponível em: https://doi.org/10.1007/978-1-4471-6684-9.

CHEN, Y.; YANG, X.-H.; WEI, Z.; HEIDARI, A. A.; ZHENG, N.; LI, Z.; CHEN, H.; HU, H.; ZHOU, Q.; GUAN, Q. Generative adversarial networks in medical image augmentation: A review. **Computers in Biology and Medicine**, Elsevier BV, v. 144, p. 105382, maio 2022. Disponível em: https://doi.org/10.1016/j.compbiomed.2022.105382.

CHOLLET, F. **Xception: Deep Learning with Depthwise Separable Convolutions**. arXiv, 2016. Disponível em: https://arxiv.org/abs/1610.02357.

CHOLLET, F. **Deep learning with Python**. 2. ed. Shelter Island, Nova York, EUA: Manning Publications, 2021. 504 p. ISBN 978-1617296864.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. IEEE, 2009. Disponível em: https://doi.org/10.1109/cvpr.2009.5206848.

FAIZAL, S.; RAJPUT, C. A.; TRIPATHI, R.; VERMA, B.; PRUSTY, M. R.; KORADE, S. S. Automated cataract disease detection on anterior segment eye images using adaptive thresholding and fine tuned inception-v3 model. **Biomedical Signal Processing and Control**, Elsevier BV, v. 82, p. 104550, abr. 2023. Disponível em: https://doi.org/10.1016/j.bspc.2022.104550.

GUO, Z.; SHEN, Y.; WAN, S.; SHANG, W.-L.; YU, K. Hybrid intelligence-driven medical image recognition for remote patient diagnosis in internet of medical things. **IEEE Journal of**

Biomedical and Health Informatics, Institute of Electrical and Electronics Engineers (IEEE), v. 26, n. 12, p. 5817–5828, dez. 2022. Disponível em: https://doi.org/10.1109/jbhi.2021.3139541.

HARDALAÇ, F.; UYSAL, F.; PEKER, O.; ÇIÇEKLIDAĞ, M.; TOLUNAY, T.; TOKGöZ, N.; KUTBAY, U.; DEMIRCILER, B.; MERT, F. Fracture detection in wrist x-ray images using deep learning-based object detection models. **Sensors**, MDPI AG, v. 22, n. 3, p. 1285, fev. 2022. Disponível em: https://doi.org/10.3390/s22031285.

HE, K.; GAN, C.; LI, Z.; REKIK, I.; YIN, Z.; JI, W.; GAO, Y.; WANG, Q.; ZHANG, J.; SHEN, D. Transformers in medical image analysis. **Intelligent Medicine**, Elsevier BV, v. 3, n. 1, p. 59–78, fev. 2023. Disponível em: https://doi.org/10.1016/j.imed.2022.07.002.

HE, K.; ZHANG, X.; REN, S.; SUN, J. **Identity Mappings in Deep Residual Networks**. arXiv, 2016. Disponível em: https://arxiv.org/abs/1603.05027.

HEILIGER, L.; SEKUBOYINA, A.; MENZE, B.; EGGER, J.; KLEESIEK, J. Beyond medical imaging - a review of multimodal deep learning in radiology. Institute of Electrical and Electronics Engineers (IEEE), fev. 2022. Disponível em: https://doi.org/10.36227/techrxiv.19103432.v1.

HEYDARIAN, M.; DOYLE, T. E.; SAMAVI, R. MLCM: Multi-label confusion matrix. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 10, p. 19083–19095, 2022. Disponível em: https://doi.org/10.1109/access.2022.3151048.

HU, M.; PAN, S.; LI, Y.; YANG, X. **Advancing Medical Imaging with Language Models: A Journey from N-grams to ChatGPT**. arXiv, 2023. Disponível em: https://arxiv.org/abs/2304.04920.

HUANG, G.; LIU, Z.; MAATEN, L. van der; WEINBERGER, K. Q. **Densely Connected Convolutional Networks**. arXiv, 2016. Disponível em: https://arxiv.org/abs/1608.06993.

HUANG, J.-H.; WU, T.-W.; YANG, C.-H. H.; WORRING, M. Deep context-encoding network for retinal image captioning. In: **2021 IEEE International Conference on Image Processing (ICIP)**. IEEE, 2021. Disponível em: https://doi.org/10.1109/icip42928.2021.9506803.

JAISWAL, A.; GIANCHANDANI, N.; SINGH, D.; KUMAR, V.; KAUR, M. Classification of the COVID-19 infected patients using DenseNet201 based deep transfer learning. **Journal of Biomolecular Structure and Dynamics**, Informa UK Limited, v. 39, n. 15, p. 5682–5689, jul. 2020. Disponível em: https://doi.org/10.1080/07391102.2020.1788642.

KAKKAR, B.; GOYAL, M.; JOHRI, P.; KUMAR, Y. Artificial intelligence-based approaches for detection and classification of different classes of malaria parasites using microscopic images: A systematic review. **Archives of Computational Methods in Engineering**, Springer Science and Business Media LLC, jun. 2023. Disponível em: https://doi.org/10.1007/s11831-023-09959-0.

KHALIFA, N. E.; LOEY, M.; MIRJALILI, S. A comprehensive survey of recent trends in deep learning for digital images augmentation. **Artificial Intelligence Review**, Springer Science and Business Media LLC, v. 55, n. 3, p. 2351–2377, set. 2021. Disponível em: https://doi.org/10.1007/s10462-021-10066-4.

KHURANA, D.; KOLI, A.; KHATTER, K.; SINGH, S. Natural language processing: state of the art, current trends and challenges. **Multimedia Tools and Applications**, Springer

Science and Business Media LLC, v. 82, n. 3, p. 3713–3744, jul. 2022. Disponível em: https://doi.org/10.1007/s11042-022-13428-4.

KIM, H. E.; COSA-LINAN, A.; SANTHANAM, N.; JANNESARI, M.; MAROS, M. E.; GANSLANDT, T. Transfer learning for medical image classification: a literature review. **BMC Medical Imaging**, Springer Science and Business Media LLC, v. 22, n. 1, abr. 2022. Disponível em: https://doi.org/10.1186/s12880-022-00793-7.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. arXiv, 2014. Disponível em: https://arxiv.org/abs/1412.6980.

KORA, P.; OOI, C. P.; FAUST, O.; RAGHAVENDRA, U.; GUDIGAR, A.; CHAN, W. Y.; MEENAKSHI, K.; SWARAJA, K.; PLAWIAK, P.; ACHARYA, U. R. Transfer learning techniques for medical image analysis: A review. **Biocybernetics and Biomedical Engineering**, Elsevier BV, v. 42, n. 1, p. 79–107, jan. 2022. Disponível em: https://doi.org/10.1016/j.bbe.2021.11.004.

LEE, H.; CHO, H.; PARK, J.; CHAE, J.; KIM, J. Cross encoder-decoder transformer with global-local visual extractor for medical image captioning. **Sensors**, MDPI AG, v. 22, n. 4, p. 1429, fev. 2022. Disponível em: https://doi.org/10.3390/s22041429.

LIU, Z.; LV, Q.; YANG, Z.; LI, Y.; LEE, C. H.; SHEN, L. **Recent Progress in Transformer-based Medical Image Analysis**. arXiv, 2022. Disponível em: https://arxiv.org/abs/2208.06643.

LOPEZ-MATTEI, J.; YANG, E. H.; BALDASSARRE, L. A.; AGHA, A.; BLANKSTEIN, R.; CHOI, A. D.; CHEN, M. Y.; MEYERSOHN, N.; DALY, R.; SLIM, A.; ROCHITTE, C.; BLAHA, M.; WHELTON, S.; DZAYE, O.; DENT, S.; MILGROM, S.; KY, B.; ILIESCU, C.; MAMAS, M. A.; FERENCIK, M. Cardiac computed tomographic imaging in cardio-oncology: An expert consensus document of the society of cardiovascular computed tomography (SCCT). endorsed by the international cardio-oncology society (ICOS). **Journal of Cardiovascular Computed Tomography**, Elsevier BV, v. 17, n. 1, p. 66–83, jan. 2023. Disponível em: https://doi.org/10.1016/j.jcct.2022.09.002.

LOSHCHILOV, I.; HUTTER, F. **Decoupled Weight Decay Regularization**. arXiv, 2017. Disponível em: https://arxiv.org/abs/1711.05101.

LU, J. **Gradient Descent, Stochastic Optimization, and Other Tales**. arXiv, 2022. Disponível em: https://arxiv.org/abs/2205.00832.

LYNDON, D.; KUMAR, A.; KIM, J. Neural captioning for the imageclef 2017 medical image challenges. In: **Conference and Labs of the Evaluation Forum**. [*S. l.*: *s. n.*], 2017.

MATSOUKAS, C.; HASLUM, J. F.; SORKHEI, M.; SöDERBERG, M.; SMITH, K. **What Makes Transfer Learning Work For Medical Images: Feature Reuse amp; Other Factors**. arXiv, 2022. Disponível em: https://arxiv.org/abs/2203.01825.

MEEM, R. F.; HASAN, K. T. **Bone Marrow Cytomorphology Cell Detection using InceptionResNetV2**. arXiv, 2023. Disponível em: https://arxiv.org/abs/2305.05430.

METAXAS, V. I.; DIMITROUKAS, C. P.; EFTHYMIOU, F. O.; ZAMPAKIS, P. E.; PANAYIOTAKIS, G. S.; KALOGEROPOULOU, C. P. Patient dose in CT angiography examinations: An institutional survey. **Radiation Physics and Chemistry**, Elsevier BV, v. 195, p. 110083, jun. 2022. Disponível em: https://doi.org/10.1016/j.radphyschem.2022.110083.

ORTIZ-TORO, C.; GARCÍA-PEDRERO, A.; LILLO-SAAVEDRA, M.; GONZALO-MARTÍN, C. Automatic detection of pneumonia in chest x-ray images using textural features. **Computers in Biology and Medicine**, Elsevier BV, v. 145, p. 105466, jun. 2022. Disponível em: https://doi.org/10.1016/j.compbiomed.2022.105466.

PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W.-J. BLEU. In: **Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02**. Association for Computational Linguistics, 2001. Disponível em: https://doi.org/10.3115/1073083.1073135.

PARK, H.; KIM, K.; PARK, S.; CHOI, J. Medical image captioning model to convey more details: Methodological comparison of feature difference generation. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 9, p. 150560–150568, 2021. Disponível em: https://doi.org/10.1109/access.2021.3124564.

PARK, H.; KIM, K.; YOON, J.; PARK, S.; CHOI, J. Feature difference makes sense: A medical image captioning model exploiting feature difference and tag information. In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop**. Association for Computational Linguistics, 2020. Disponível em: https://doi.org/10.18653/v1/2020.acl-srw.14.

PAVLOPOULOS, J.; KOUGIA, V.; ANDROUTSOPOULOS, I. A survey on biomedical image captioning. In: **Proceedings of the Second Workshop on Shortcomings in Vision and Language**. Association for Computational Linguistics, 2019. Disponível em: https://doi.org/10.18653/v1/w19-1803.

PELKA, O.; KOITKA, S.; RüCKERT, J.; NENSA, F.; FRIEDRICH, C. M. Radiology objects in COntext (ROCO): A multimodal image dataset. In: **Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis**. Springer International Publishing, 2018. p. 180–189. Disponível em: https://doi.org/10.1007/978-3-030-01364-6_20.

RAJ, R. J. S.; SHOBANA, S. J.; PUSTOKHINA, I. V.; PUSTOKHIN, D. A.; GUPTA, D.; SHANKAR, K. Optimal feature selection-based medical image classification using deep learning model in internet of medical things. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 8, p. 58006–58017, 2020. Disponível em: https://doi.org/10.1109/access.2020.2981337.

RAMESH, R.; SATHIAMOORTHY, S. A deep learning grading classification of diabetic retinopathy on retinal fundus images with bio-inspired optimization. **Engineering, Technology &amp Applied Science Research**, Engineering, Technology & Applied Science Research, v. 13, n. 4, p. 11248–11252, ago. 2023. Disponível em: https://doi.org/10.48084/etasr.6033.

ROSTAM-ALILOU, A. A.; SAFARI, M.; JARRAH, H. R.; ZOLFAGHARIAN, A.; BODAGHI, M. A machine learning model for non-invasive detection of atherosclerotic coronary artery aneurysm. **International Journal of Computer Assisted Radiology and Surgery**, Springer Science and Business Media LLC, v. 17, n. 12, p. 2221–2229, ago. 2022. Disponível em: https://doi.org/10.1007/s11548-022-02725-w.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. arXiv, 2018. Disponível em: https://arxiv.org/abs/1801.04381.

SARVAMANGALA, D. R.; KULKARNI, R. V. Convolutional neural networks in medical image understanding: a survey. **Evolutionary Intelligence**, Springer Science and Business Media LLC, v. 15, n. 1, p. 1–22, jan. 2021. Disponível em: https://doi.org/10.1007/s12065-020-00540-3.

SCHWARTZ, F. R.; SAMEI, E.; MARIN, D. Exploiting the potential of photon-counting CT in abdominal imaging. **Investigative Radiology**, Ovid Technologies (Wolters Kluwer Health), v. 58, n. 7, p. 488–498, jan. 2023. Disponível em: https://doi.org/10.1097/rli.0000000000000949.

SHARMA, H.; PADHA, D. A comprehensive survey on image captioning: from handcrafted to deep learning-based techniques, a taxonomy and open research issues. **Artificial Intelligence Review**, Springer Science and Business Media LLC, abr. 2023. Disponível em: https://doi.org/10.1007/s10462-023-10488-2.

SHAZEER, N.; STERN, M. **Adafactor: Adaptive Learning Rates with Sublinear Memory Cost**. arXiv, 2018. Disponível em: https://arxiv.org/abs/1804.04235.

SILVA, E. L. da; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianópolis: Universidade Federal de Santa Catarina, 2005. 139 p.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. arXiv, 2014. Disponível em: https://arxiv.org/abs/1409.1556.

SINGH, A.; RAGURU, J. K.; PRASAD, G.; CHAUHAN, S.; TIWARI, P. K.; ZAGUIA, A.; ULLAH, M. A. Medical image captioning using optimized deep learning model. **Computational Intelligence and Neuroscience**, Hindawi Limited, v. 2022, p. 1–9, mar. 2022. Disponível em: https://doi.org/10.1155/2022/9638438.

SINGH, D.; KAUR, M.; ALANAZI, J. M.; ALZUBI, A. A.; LEE, H.-N. Efficient evolving deep ensemble medical image captioning network. **IEEE Journal of Biomedical and Health Informatics**, Institute of Electrical and Electronics Engineers (IEEE), v. 27, n. 2, p. 1016–1025, fev. 2023. Disponível em: https://doi.org/10.1109/jbhi.2022.3223181.

SONG, L.; HE, C.; XU, L.; ZHENG, B. CapGAN: Medical image captioning with clinical style and pathology preservation using conditional generative adversarial nets. In: **2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)**. IEEE, 2022. Disponível em: https://doi.org/10.1109/prai55851.2022.9904068.

SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V.; ALEMI, A. **Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning**. arXiv, 2016. Disponível em: https://arxiv.org/abs/1602.07261.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. IEEE, 2016. Disponível em: https://doi.org/10.1109/cvpr.2016.308.

TOĞAÇAR, M.; CöMERT, Z.; ERGEN, B. Intelligent skin cancer detection applying autoencoder, MobileNetV2 and spiking neural networks. **Chaos, Solitons &amp Fractals**, Elsevier BV, v. 144, p. 110714, mar. 2021. Disponível em: https://doi.org/10.1016/j.chaos.2021.110714.

TUNSTALL, L.; WERRA, L. V.; WOLF, T. **Natural language processing with transformers**. 1. ed. Sebastopol, Califórnia, EUA: O'Reilly Media, Inc., 2022. 406 p. ISBN 978-9355420329.

ULLAH, N.; KHAN, J. A.; KHAN, M. S.; KHAN, W.; HASSAN, I.; OBAYYA, M.; NEGM, N.; SALAMA, A. S. An effective approach to detect and identify brain tumors using transfer learning. **Applied Sciences**, MDPI AG, v. 12, n. 11, p. 5645, jun. 2022. Disponível em: https://doi.org/10.3390/app12115645.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. **Attention Is All You Need**. arXiv, 2017. Disponível em: https://arxiv.org/abs/1706.03762.

VILLARRAGA-GÓMEZ, H.; HERAZO, E. L.; SMITH, S. T. X-ray computed tomography: from medical imaging to dimensional metrology. **Precision Engineering**, Elsevier BV, v. 60, p. 544–569, nov. 2019. Disponível em: https://doi.org/10.1016/j.precisioneng.2019.06.007.

WITHERS, P. J.; BOUMAN, C.; CARMIGNATO, S.; CNUDDE, V.; GRIMALDI, D.; HAGEN, C. K.; MAIRE, E.; MANLEY, M.; PLESSIS, A. D.; STOCK, S. R. X-ray computed tomography. **Nature Reviews Methods Primers**, Springer Science and Business Media LLC, v. 1, n. 1, fev. 2021. Disponível em: https://doi.org/10.1038/s43586-021-00015-4.

WU, T.-W.; HUANG, J.-H.; LIN, J.; WORRING, M. Expert-defined keywords improve interpretability of retinal image captioning. In: **2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)**. IEEE, 2023. Disponível em: https://doi.org/10.1109/wacv56688.2023.00190.

XU, M.; YOON, S.; FUENTES, A.; PARK, D. S. A comprehensive survey of image augmentation techniques for deep learning. **Pattern Recognition**, Elsevier BV, v. 137, p. 109347, maio 2023. Disponível em: https://doi.org/10.1016/j.patcog.2023.109347.

ZEILER, M. D. **ADADELTA: An Adaptive Learning Rate Method**. arXiv, 2012. Disponível em: https://arxiv.org/abs/1212.5701.

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q. V. **Learning Transferable Architectures for Scalable Image Recognition**. arXiv, 2017. Disponível em: https://arxiv.org/abs/1707.07012.

ZORIĆ, B.; MATIĆ, T.; HOCENSKI, Ž. Classification of biscuit tiles for defect detection using fourier transform features. **ISA Transactions**, Elsevier BV, v. 125, p. 400–414, jun. 2022. Disponível em: https://doi.org/10.1016/j.isatra.2021.06.025.