INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA
E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS - DISTRITO INDUSTRIAL
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

ROMÃO CHARLES SILVA E SILVA

PREVENÇÃO CONTRA DEEPFAKES: DESENVOLVIMENTO DE UM SISTEMA DE RECONHECIMENTO FACIAL PARA DIFERENCIAR ROSTOS HUMANOS DE ROSTOS GERADOS POR IA

MANAUS - AM

2024

ROMÃO CHARLES SILVA E SILVA

PREVENÇÃO CONTRA DEEPFAKES: DESENVOLVIMENTO DE UM SISTEMA DE RECONHECIMENTO FACIAL PARA DIFERENCIAR ROSTOS HUMANOS DE ROSTOS GERADOS POR IA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Controle e Automação, do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial – IFAM/CMDI.

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS - AM

2024

Dados Internacionais de Catalogação na Publicação (CIP)

S586p

Silva, Romão Charles Silva e.
Prevenção contra *deepfakes*: desenvolvimento de um sistema de reconhecimento facial para diferenciar rostos humanos de rostos gerados por IA / Romão Charles Silva e Silva. — Manaus, 2024.
70f.: il. color.

Monografia (Graduação) — Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, Curso de Engenharia de Controle e Automação, 2024.
Orientador: Prof.º Alyson de Jesus dos Santos, Dr.

1. Inteligência artificial. 2. Tensorflow. 3. Imagens geradas. I. Santos, Alyson de Jesus dos. II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.895

Elaborada por Oziane Romualdo de Souza (CRB11/ nº 734)

ANEXO 7

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 19 dias do mês de Março, de 2024, às 18:30 h, o(a) discente Ronan Charles Silva e Silva apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Dr. Alyson de Jesus dos Santos (docente-orientador), Prof(a). _____ (coorientador), Prof(a). Dr. Gilbert Breves Martins (Membro 1) e Prof(a). Msc. Marlos André Silva Rodrigues (Membro 2). A sessão publica de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título Avaliador de Autenticidade Facial: Desenvolvimento de um sistema de Reconhecimento Facial para Diferenciar Rostos Humanos e Rostos Gerados por IA. Na sequencia, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final Oito, Quatro (8,4) do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 19/04/2024, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às ____h ____ min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alyson de Jesus dos Santos

Prof.(a) Avaliador 1: Gilbert Breves Martins

Prof.(a) Avaliador 2: _____

Discente: Ronan Charles Silva e Silva

AGRADECIMENTOS

Primeiramente agradeço a Deus, por ter me dado forças e motivação desde o início dessa graduação através de meus pais e minha avó, que durante esses anos foram fonte de motivação, suporte e encorajamento.

Também agradeço as minhas irmãs, noiva e amigos, por estarem presentes comigo nessa difícil trajetória, o seu suporte veio de diferentes maneiras através dos anos e sou muito grato a todos vocês. Agradeço aos meus professores por todo o seu conhecimento transmitido, e agradeço grandemente ao meu orientador, Prof. Dr. Alyson de Jesus do Santos, que sem ele eu ainda não teria certeza do que fazer para o trabalho final e não saberia como prosseguir.

“Na vida, ao contrário do xadrez, o jogo continua após o xeque-mate.”

-Isaac Asimov

RESUMO:

Esse trabalho tem por objetivo o desenvolvimento de um sistema baseado em aprendizado de máquina que diferencie imagens de pessoas geradas por inteligência artificial de pessoas reais, capacidade que pode ser muito útil para a identificação de golpes que usam imagens geradas. O desenvolvimento do projeto foi feito em 3 passos principais: organização dos dados, treinamento e teste. O sistema foi inteiramente feito no Google Colab, portanto, utilizou Python sendo a principal ferramenta de desenvolvimento o Tensorflow, foram treinados dois modelos, sendo que um tem quase o dobro de imagens usadas para treinamento, com a intenção de observar as consequências de usar um conjunto maior. Ao final do projeto são mostrados os resultados quantitativos e qualitativos da classificação de imagens.

Palavras-chave: Inteligência Artificial; Tensorflow; Imagens Geradas.

ABSTRACT:

This work aims to develop a system based on machine learning that differentiates images of people generated by artificial intelligence from real people, a capability that can be very useful for identifying scams that use generated images. The development of the project was done in 3 main steps: data organization, training and testing. The system was entirely made in Google Colab, therefore, it used Python and the main development tool Tensorflow, two models were trained, one of which has almost twice as many images used for training, with the intention of observing the consequences of using a larger set. At the end of the project, the quantitative and qualitative results of the image classification are shown.

Keywords: Artificial Intelligence; Tensorflow; Generated Images.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AM	Aprendizado de Máquina (*Machine Learning*)
AUC	*Area Under the ROC Curve*
CNN	*Convolutional Neural Network*
DL	Aprendizado Profundo *(Deep Learning)*
IA	Inteligência Artificial
IoU	Intersecção de União (*Intersection of Union*)
GAN	Rede Adversária Generativa (*Generative Adversarial Network)*
GPU	Unidade de Processamento Gráfico (*Graphic Processing Unit*)
MVS	Máquinas de Vetores de Suporte
TCC	Trabalho de Conclusão de Curso
TPU	*Tensor Processor Unit*
RNA	Redes Neurais Artificiais
RNC	Rede Neural Convolucional
RNG	Rede Neural Generativa
ROC	*Receiver Operating Characteristic*

SUMÁRIO

1 Introdução

O número de perfis falsos em redes sociais representa uma grande parte da quantidade total de contas, em geral são feitos para aplicarem golpes e espalhar desinformação, segundo os dados obtidos por (MOORE, 2023) em 2012 o Facebook admitiu que 83 milhões de contas de sua plataforma eram falsas ou duplicadas, esse número subiu para 260 milhões em 2017, quantidade essa que segundo o IBGE superava a população brasileira que era de 207,7 milhões, em 2020 o Facebook retornou com um reporte avaliando que 448 milhões das contas eram falsas ou duplicadas.

Segundo a PSafe, uma empresa de cibersegurança, 1 a cada 4 pessoas que se relacionaram com outras pessoas pela web foram vítimas de perfis falsos, o estudo ainda revelou que 25,5% das vítimas tiveram perdas financeiras (BRÊTAS, 2022).

Com os avanços recentes da Inteligência Artificial como ferramenta para a geração imagens, o número de criação de perfis falsos tendem a aumentar, anteriormente um perfil falso precisava da foto de uma pessoa real para ter mais confiabilidade como foi o caso de um brasileiro que descobriu mais 100 perfis falsos com a sua foto (TAGIAROLI, 2023), porém com as novas ferramentas de geração de imagens essa barreira está sendo superada, golpistas podem pedir por detalhes específicos como gênero, idade, cor e tipo de cabelo para a IA ao gerar a imagem de uma pessoa.

O projeto propõe o desenvolvimento de um sistema que se baseia em Inteligência Artificial, que será apto a extrair características de autênticos rostos humanos e rostos gerados por inteligência artificial para que seja capaz de identificar e diferenciar.

1.1 Justificativa

A proliferação de *deepfakes*, que são imagens e vídeos realistas criados por IA, é uma preocupação crescente que ameaça a veracidade do conteúdo online. Este trabalho tem como objetivo contribuir para a mitigação desses conteúdos prejudiciais.

A proteção da privacidade e dos direitos individuais é a segunda motivação para este estudo. Com o progresso das tecnologias de IA, a criação não consentida de imagens realistas de pessoas reais tornou-se uma possibilidade perigosa. Isso pode resultar em sérios problemas de privacidade e consentimento, pois essas imagens podem ser usadas de maneira inapropriada ou mal-intencionada.

O sistema proposto tem como objetivo detectar essas imagens geradas por IA assim contribuindo para a proteção da privacidade, direitos individuais como também a prevenção de crimes virtuais. Ao identificar e marcar essas imagens, pode-se prevenir o uso indevido de imagens que referenciam pessoas reais e garantindo sua integridade.

Em resumo, este Trabalho de Conclusão de Curso (TCC) visa o desenvolvimento de uma ferramenta importante na luta contra a disseminação de *deepfakes* e rostos gerados por IA, para que assim possa contribuir na proteção da privacidade na era digital. Este trabalho demonstra o papel vital que a visão computacional e a IA podem desempenhar na solução de desafios éticos e de segurança na sociedade contemporânea.

1.2 Objetivos Gerais

Diante do cenário apresentado, a realização deste TCC tem como objetivo desenvolver um sistema de reconhecimento facial capaz de diferenciar os rostos genuinamente humanos de rostos gerados por inteligência artificial, para que dessa forma torne-se um sistema adequado para a prevenção de *deepfakes* e rostos gerados por IA comumente usados para crimes virtuais.

1.3 Objetivos específicos

Para se julgar a eficiência do projeto serão estabelecidos os seguintes objetivos específicos:

a) Desenvolver um banco de dados robusto, capaz de servir como base para o treinamento do sistema.
b) Configurar o ambiente ideal para o desenvolvimento do projeto.
c) Treinar o sistema com o banco de imagens desenvolvido.
d) Testar e comparar o funcionamento do sistema.

1.4 Metodologia

Por se tratar de um projeto que envolve pesquisa e desenvolvimento, é certo que a produção do sistema que tem como função identificar e diferenciar rostos reais de rostos gerados por inteligência artificial deve seguir uma metodologia organizada, sendo essa:

a) Levantamento Bibliográfico.
b) Coleta e Verificação de dados.
c) Treinamento do Sistema.
d) Teste do Sistema.
e) Avaliação dos Resultados

2 Referencial Teórico

2.1 Inteligência Artificial

A Inteligência Artificial (IA) é um dos conceitos mais importantes de serem definidos no contexto desse TCC, sendo ela uma das tecnologias com maior evolução dos últimos anos trata-se de uma área extremamente ampla com diversas aplicações para todas as áreas de atividade e conhecimento humano.

2.1.1 Definição

Segundo a definição de D'Addario (2022) a IA é uma disciplina que desenvolve modelos computacionais com a capacidade de exibir comportamento inteligente ou racional, tendo como propósito de facilitar e automatizar a resolução de problemas em diversas áreas do conhecimento. Outra definição também abrange a habilidade de adquirir, compreender, pensar, raciocinar e aplicar conhecimento.

Devido a sua rápida evolução, sua definição também pode mudar ao longo do tempo, é comum que as definições mais recentes enfatizem o "imitar o comportamento humano inteligente", representando um conceito mais robusto. Há um tempo considerável, a comunidade de Inteligência Artificial tem se dedicado a replicar comportamentos inteligentes por meio de programas de computador.

2.1.2 Origem

Sua origem remete a tentativa de criar autômatos, que simulem habilidades dos seres humanos, mas a origem do conceito se deve ao matemático Alan Turing e o termo ao McCarthy, que se reuniram com pesquisadores em 1956 no Darmouth College (Estados Unidos), para discutir a probabilidade de construir máquinas que não se limitam

a realizar cálculos predeterminados, mas sim, operações “inteligentes” (D'ADDARIO, 2022).

Foi Alan Turing quem propôs o famoso Teste de Turing, também conhecido como Jogo da Imitação, esse teste tem como objetivo saber se uma máquina consegue se passar por um ser humano sem ser descoberta, o teste é constituído por um computador e duas pessoas, uma delas sendo quem vai conversar com a máquina enquanto a outra será o juiz, a conversa deverá ser feita em texto, a máquina deve ser capaz de demonstrar inteligência e sensibilidade como um ser humano faria, ao final da conversa o juiz deve dar um veredito para definir quem é a máquina e quem é o ser humano.

2.1.3 Evolução e Aplicação

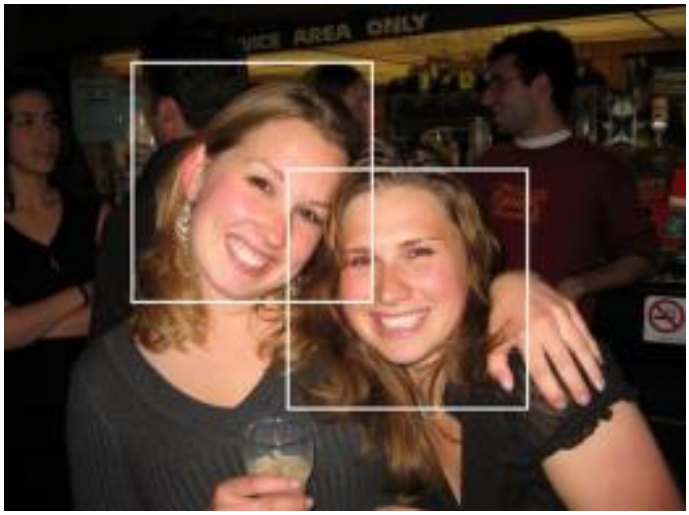
Os avanços da IA já estão presentes no dia a dia de grande parte da população, estão presentes nos tradutores e teclados dos smartfones graças ao processamento de linguagem natural, os algoritmos de recomendação dos serviços de streaming que visam sempre deixar as pessoas consumirem cada vez mais o tipo de conteúdo que lhes agrada.

Conforme mencionado por Ludermir (2021), a Inteligência Artificial permitiu que as máquinas, que anteriormente desempenhavam predominantemente tarefas manuais, passassem a realizar atividades racionais, como a condução de carros, exemplificada pelos desenvolvimentos da Google e Tesla, nos quais os veículos podem se movimentar de forma autônoma, sem a necessidade de um motorista.

A plena autonomia desses sistemas aguarda apenas a regulamentação apropriada e testes em situações reais, pois é necessário possuir uma robustez adequada para lidar com situações não previamente treinadas, incluindo aquelas que representem potenciais riscos para as vidas de motoristas e pedestres, conforme mencionado por Feng *et al.* (2021).

Uma aplicação amplamente reconhecida e bem-sucedida é a implementada pelo Facebook, exemplificada na **Figura** 1, onde, por meio do uso de Redes Neurais Profundas, foi desenvolvido o DeepFace. Esse sistema é capaz de identificar rostos humanos em imagens digitais, independentemente do ângulo de visualização.

Figura 1 - Identificação de face.



Fonte: Stone *et al.* (2008)

Grande parte da atual evolução da IA se deve também a evolução do processamento computacional, que permite um maior gerenciamento de dados e testes de novas arquiteturas. Como esclarecido por Aradi (2020) as Unidades de Processamento Gráfico (GPU), atualmente mais poderosas, são capazes de lidar eficientemente com grandes volumes de dados, atendendo a uma exigência fundamental dos sistemas de Inteligência Artificial, uma vez que estes são fortemente baseados em dados.

2.2 Aprendizado de Máquina (AM)

O aprendizado de máquina surgiu como campo da inteligência artificial na década de 60 que desenvolve aprendizado através de dados. No final da década de 90, essa área se expandiu a ponto de estabelecer como um campo por si mesma. Em específico,

as aplicações de AM começaram a ter uma intersecção com as de estatística (IZBICKI e DOS SANTOS, 2020).

As técnicas de AM são orientadas a dados, isto é, aprendem automaticamente a partir de grandes volumes de dados. Os algoritmos de AM geram hipóteses a partir dos dados (LUDERMIR, 2021). Segundo (IZBICKI e DOS SANTOS, 2020) até os anos 90 a comunidade estatística começou a incorporar métodos em AM, por outro lado o recente fortalecimento de AM fez a direção oposta se tornar mais comum, que são os métodos desenvolvidos por AM começarem a ser usados em estatística.

Suas aplicações são multidisciplinares, sendo aplicados em finanças, detecção de fraude, manufatura, medicina, telecomunicações, robótica, reconhecimento de fala, pessoas ou objetos (ALPAYDIN, 2020).

Existem quatro principais categorias de aprendizado de máquina: por reforço, supervisionado, não supervisionado e semi-supervisionado.

No aprendizado por reforço, o algoritmo é exposto à resposta correta e a um sinal de reforço, que pode ser uma recompensa ou punição. O algoritmo formula uma hipótese com base nos exemplos fornecidos e determina se a detecção foi considerada boa ou ruim (LUDEMIR, 2021). Esses algoritmos são equipados com um agente que interage com um ambiente, aprendendo por tentativa e erro a melhor forma de se comportar para otimizar o desempenho. O agente deve perceber o estado do ambiente e, a partir dessa percepção, tomar ações que levem a um novo estado (ARADI, 2020).

Como apontado por Calanca *et al.* (2023), no aprendizado supervisionado os computadores são treinados usando exemplos de entrada e as saídas correspondentes. O algoritmo aprende a associar padrões e relações entre esses exemplos, permitindo que, posteriormente, possa realizar previsões ou classificações em novos conjuntos de dados com base no treinamento recebido. Esse método é conhecido como aprendizado supervisionado.

No aprendizado não supervisionado, os dados são apresentados sem rótulos, e o propósito do algoritmo é agrupar esses dados com base em suas similaridades, analisando seus atributos. Após a formação dos agrupamentos, é essencial examinar o significado de cada grupo no contexto do problema em questão (LUDEMIR, 2021).

Conforme apontado por Zhu (2005) o aprendizado semi-supervisionado ao combinar dados rotulados e não rotulados, busca reduzir a intervenção humana no algoritmo e aproveita a abundância de dados não rotulados disponíveis. No entanto, desafios incluem a possibilidade de rótulos de baixa qualidade, que podem prejudicar a precisão da classificação, e a complexidade em identificar classificações de qualidade inferior que podem impactar negativamente o sistema desenvolvido.

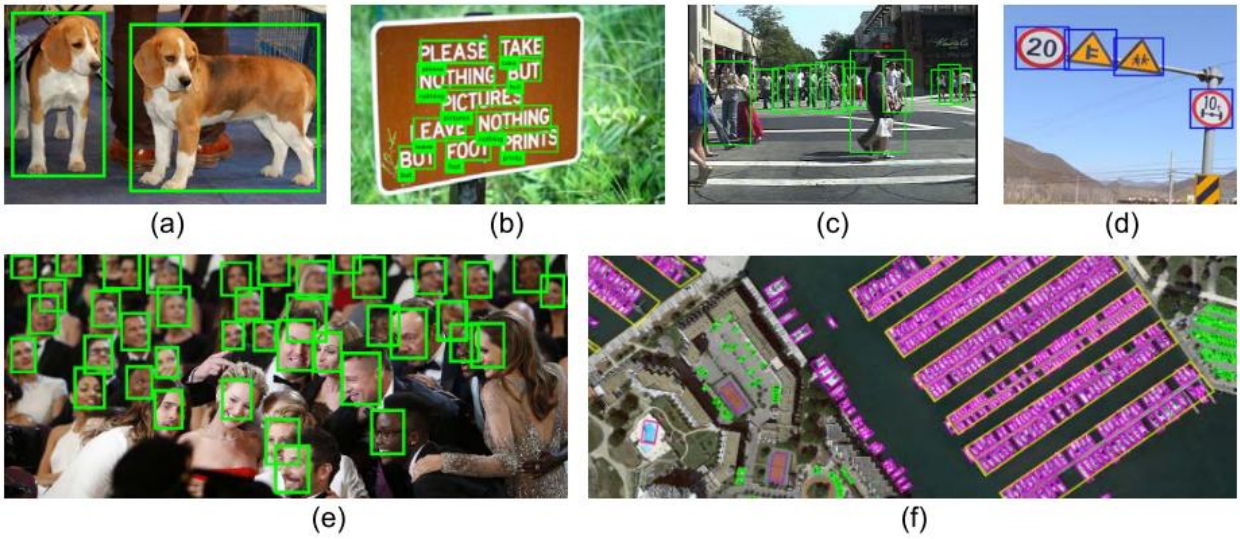
2.3 Visão Computacional

Uma das formas de um ser humano obter informações é através da visão, a imagem é obtida pelos olhos para ser processada pelo cérebro, sendo assim reconhecida ou associada a algum conceito significativo. Uma imagem para um computador, sendo uma foto ou vídeo, trata-se de uma matriz de vários pixels, fazer com a que a máquina entenda o que está vendo é desafio comum para a Visão Computacional, pois, segundo Li *et al.* (2020) a máquina deve reconhecer a que conceito uma determinada parte dos dados obtidos representa nos dados armazenados.

A visão computacional é um dos ramos da computação que tem experimentado um crescimento mais notável em anos recentes. Tanto a detecção de rostos quanto de objetos em cenas são dois exemplos claros de aplicações de visão artificial com maior impacto. Não deve passar despercebido que são ferramentas que constantemente desempenham funções de análise, interpretação, e manipulação de imagens (SIGUT *et al.*, 2020). Com tal capacidade, as possibilidades para suas aplicações tornam-se praticamente incontáveis, sendo possível a sua implementação em todas as práticas

humanas, principalmente no contexto de segurança seja por detecção e monitoramento de pessoas e objetos.

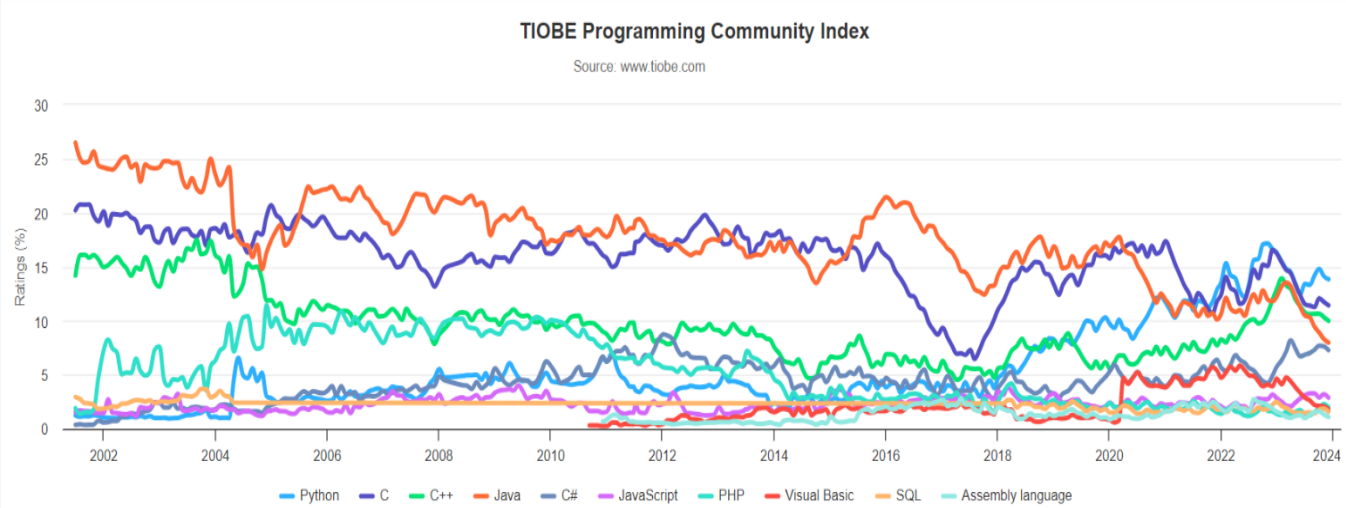
Figura 2 -Diversas aplicações para visão computacional: a) detecção genérica de objeto, b) detecção de texto, c) detecção de pedestres, d) detecção de sinais de trânsito, e) detecção de faces, f) detecção de objetos em imagens aéreas.



Fonte: AGARWAL *et al.* (2019)

Existem diversas bibliotecas que contribuem para os avanços para as pesquisas nessa área, constituídos por arquiteturas e algoritmos para o processamento de imagens, a mais conhecida é a *Open Source Computer Vision Library* (OpenCV), trata-se de uma biblioteca *open-source* que pode ser usar em diversas plataformas: Windows, Linux, MacOS e Android. Segundo Chandrakala *et al.* (2022) o uso de OpenCV em conjunto com Python atraem mais a atenção dos pesquisadores do campo de reconhecimento devido a sua simplicidade, principalmente porque com o uso do Python pode ser mais facilmente implementado. No gráfico abaixo é mostrado o ranking de popularidade do Python em comparação a outras linguagens.

Figura 3 - Ranking de popularidade feito pelo TIOBE, empresa especializada em avaliar softwares.



Fonte: NAKAGAWA (2022)

Como mostrado no gráfico acima a linguagem de programação mais usada é Python, nos últimos anos a sua procura para o desenvolvimento de projetos cresceu bastante a ponto de superar a linguagem JAVA, uma linguagem que por vários anos consecutivos dominou a primeira posição. Segundo Silva (2023), o Python é uma linguagem amplamente usada no campo da IA por alguns motivos:

- Simplicidade e Legibilidade: Isso significa que é uma linguagem de fácil entendimento, algo que facilita as etapas de desenvolvimento e manutenção do código, características fundamentais para projetos complexos.
- Número de bibliotecas: São diversas as bibliotecas e frameworks de IA que facilitam a implementação de algoritmos de IA.
- Literatura e comunidade ativa: São diversas as literaturas em que os processos foram feitos em Python que contribuem para comunidade de desenvolvedores.

Devido a essas características o Python tornou-se a linguagem escolhida para o desenvolvimento do projeto.

2.4 Detecção de Objetos

A detecção de objetos é um campo importante no domínio da visão computacional. Várias modelos de *machine learning* (ML) e *deep learning* (DL) são empregados para melhorar o desempenho no processo de detecção de objetos e tarefas relacionadas.

Antigamente, detectores de objeto de dois estágios eram bastante populares e eficazes (DIWAN *et al.*, 2023). Como demonstrado no extenso estudo de Agarwal, Terrail e Jurie (2019) são diversas as aplicações da detecção de objetos e incluem campos como direção autônoma, identificação de objetos no ar, detecção de texto, vigilância, operações de resgate, robótica, reconhecimento facial, identificação de pedestres, motores de busca visual, cálculo de objetos de interesse, detecção de marcas e muitos outros.

Classificação de imagens é uma tarefa de classificar uma imagem ou um objeto em uma imagem em uma das categorias pré-definidas. Esse problema é geralmente resolvido com a ajuda de algoritmos de aprendizado de máquina supervisionado ou de aprendizado profundo, nos quais o modelo é treinado em um grande conjunto de dados rotulado. Alguns dos modelos de aprendizado de máquina mais usados para essa tarefa incluem Redes Neurais Artificiais (RNA), Máquina de Vetores de Suporte (MVS), Árvores de decisão e algoritmo KNN (THAI *et al.*, 2012).

No entanto, no lado do *Deep Learning* (DL), as Redes Neurais Convolucionais (RNC) e seus sucessores arquitetônicos e variantes dominam outros modelos profundos para classificar imagens e trabalhos relacionados. Além de modelos bem definidos de aprendizado de máquina e aprendizado profundo, também se pode testemunhar o uso de outras abordagens, como lógica Fuzzy e Algoritmos Genéticos para as tarefas mencionadas (GAVALI, 2019).

A localização de objetos é a operação de determinar a posição de um objeto ou de múltiplos objetos em uma imagem/frame com a ajuda de uma caixa retangular ao redor de um objeto, comumente conhecido como caixa de contorno. No entanto, a segmentação de imagem é o processo de particionar a imagem em múltiplos segmentos no qual o segmento pode conter um objeto completo ou parte de um objeto. A Segmentação de imagem é muito utilizada para localizar objetos, linhas e curvas e limites, um exemplo claro está na **Figura 4**.

Figura 4 - Exemplo de localização.

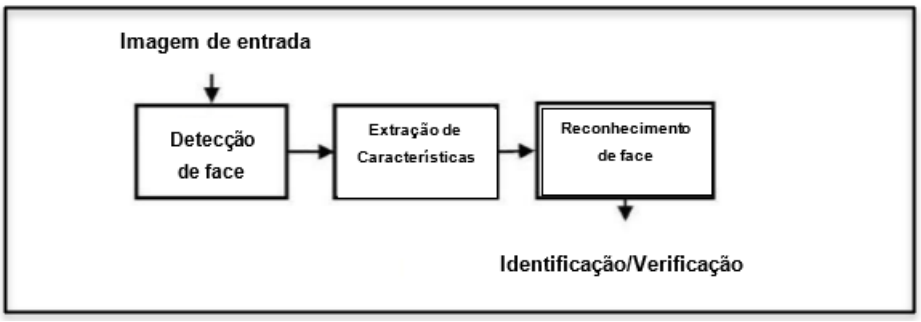
Fonte: AGARWAL *et al.* (2019)**.**

2.5 Reconhecimento de Faces

Como o foco do trabalho será o reconhecimento e classificação de faces, o objeto a ser reconhecido será o de faces humanas. O reconhecimento facial é um dos campos mais promissores em visão computacional. Foi popularizado graças a nossa cultura popular no *mainstream*, o que também levou à sua implementação em vários graus (JAGTAP *et al.*, 2019).

Atualmente são diversos os algoritmos que realizam a o reconhecimento facial, todos eles devem seguir pelo menos 4 passos para que obtenham resultados satisfatórios, os passos são: obtenção da imagem, detecção facial, extração das características e o reconhecimento facial como mostrado na **Figura** 5.

Figura 5 - Etapas para o reconhecimento de faces



Fonte: BAKSHI e SINGHAL (2014).

É importante que os passos mostrados sejam bem definidos:

1. Imagem de entrada: Segundo Rudek (2001) o processo de aquisição de uma imagem para um sistema de visão computacional envolve dois elementos: o hardware da câmera e o programa, sendo o programa quem assume a responsabilidade pelo processamento da imagem e pela coordenação das ações a serem executadas.
2. Detecção de faces: O reconhecimento de faces é uma das habilidades que os seres humanos conseguem desenvolver desde muito cedo, os estudos para implementação dessa capacidade nos computadores remontam desde os anos 1970 como o trabalho de Kelly (1970). Segundo a pesquisa de Yang *et al.* (2016) os algoritmos de detecção moderna facial moderna podem ser divididos em 4 grupos: métodos baseados em cascata, métodos baseados em partes, método baseado em recurso de canal e métodos baseados em redes neurais. E com o passar do tempo o desenvolvimento de algoritmos de reconhecimento facial vão ganhando mais atenção devido a abundância de dados disponíveis (ZHOU, 2017).
3. Extração de características: Nessa etapa o principal objetivo é a extração das características das imagens faciais identificadas na fase de detecção. Durante

esse processo, um rosto é representado por um conjunto de vetores de características denominado "assinatura", que detalha os aspectos proeminentes da imagem, como boca, nariz e olhos, acompanhados de sua distribuição geométrica (SMACH, 2007). A identificação facial é realizada com base nas características estruturais, tamanho e forma de cada rosto. Diversas técnicas são empregadas para essa identificação, que incluem a extração da forma da boca, olhos ou nariz, levando em consideração aspectos como tamanho e distância (NAPOLÉON, 2017). Existem vários métodos para a extração de características de imagens, conforme descrito por (KORTLI *et al.*, 2020), estes incluem o Histograma de Gradiente Orientado (HOG), citado por WANG *et al.* (2019), a análise de componentes independentes (ICA), a análise discriminante linear (LDA) de Annalakshmi *et al.* (2019), a transformada de características invariante de escala (SIFT) usada por Vinay (2015), o filtro de Gabor, a quantização de fase local (LPQ), as onduletas de Haar, as transformadas de Fourier (SMACH, 2007) e o padrão binário local (LBP) como usado por Napoléon (2017). Esses métodos são amplamente utilizados para a extração de características faciais.

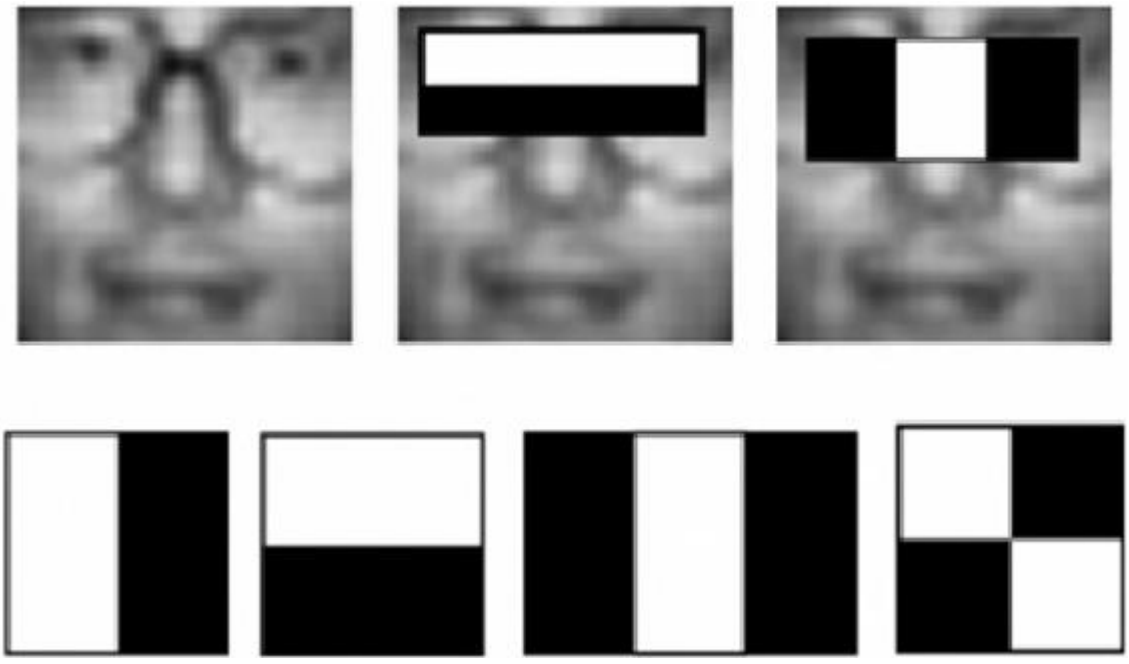
4. Reconhecimento da face: Essa etapa leva em consideração as características extraídas no processo anterior e as compara com as faces armazenadas em um banco de dados específico. Segundo Kortli *et al.* (2020) existem duas aplicações gerais para o reconhecimento de faces, uma é a identificação outra a verificação. Durante o processo de identificação a face testada é comparada com um conjunto de rostos com o objetivo de encontrar qual mais provavelmente coincide. Durante o processo de verificação a face testada é comparada com um rosto conhecido da base de dados para a tomada de decisão, seja ela de aceitar ou rejeitar.

2.5.1 Método HaarCascade

O método Haarcascade é uma técnica fundamental no campo de visão computacional, especialmente no reconhecimento facial. Ele foi introduzido por (VIOLA e JONES, 2001), ele é amplamente utilizado para detectar objetos em imagens e vídeos, mas principalmente para a detecção de faces. O Haarcascade baseia-se em classificadores em cascata, que são compostos por múltiplos classificadores fracos. Esses classificadores utilizam características simples, conhecidas como “Haar-like features”, para identificar objetos em uma imagem.

O treinamento do modelo envolve amostras positivas (com o objeto de interesse, como faces) e negativas (sem o objeto). Durante a detecção, o modelo percorre várias etapas, rejeitando rapidamente regiões não relevantes na imagem.

Figura 6 - Introduzidas pela primeira vez em 2001, as cascatas de Haar são uma classe de algoritmos de detecção de objetos



Fonte: Rosebrock (2021).

A importância do Haarcascade para o reconhecimento facial reside em sua eficiência e robustez, ele é:

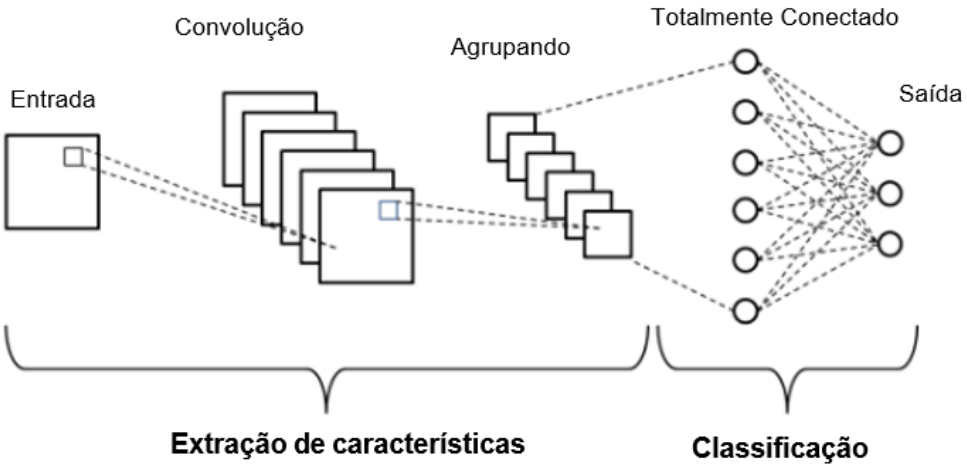
- Eficiente: O Haarcascade é computacionalmente eficiente, sendo adequado para aplicações em tempo real. Sua estrutura em cascata permite a rápida rejeição de regiões não relacionadas a faces.
- Robusto: O método é robusto a variações de iluminação, pose e escala. Ele pode detectar faces mesmo quando parcialmente ocultas ou rotacionadas.
- Amplamente Utilizado: O Haarcascade está integrado em bibliotecas de software, como o OpenCV, tornando-o acessível e popular na comunidade de visão computacional.

2.6 Rede Neural Convolucional (RNC)

O método mais popular para o reconhecimento facial envolve RNC, segundo Kelana (2021) trata-se do estado da arte do algoritmo de DL, pois RNC é uma rede neural muito semelhante as redes neurais regulares.

Uma Rede Neural Convolucional é composta por neurônios com pesos e vieses, estes que podem ser aprendidos. Cada um desses neurônios recebe valores de entrada, realizam um produto escalar e executam uma não-linearidade (se necessário). Desde os pixels brutos da imagem em uma extremidade até as pontuações de classe na outra, toda a rede ainda representa uma única função de pontuação diferenciável (KAMENCAY *et al.*, 2017).

Figura 7 - Modelo de Rede Neural Convolucional



Fonte: Kelana (2021).

A RNC é constituída por várias camadas, cada camada recebe como entrada uma matriz multidimensional de números e produz outra matriz multidimensional de números como saída, sendo essa a entrada da camada seguinte. Ao classificar imagens, a primeira camada recebe a imagem de entrada (32x32), enquanto a última camada gera um conjunto de probabilidades para cada categoria (por exemplo, 10 números, se houver dez categorias). Uma RNC básica é formada por camadas, cada uma aplicando uma função diferenciável para transformar um volume de ativações em outro (KAMENCAY *et al.*, 2017).

2.7 Redes Adversárias Generativas (GAN)

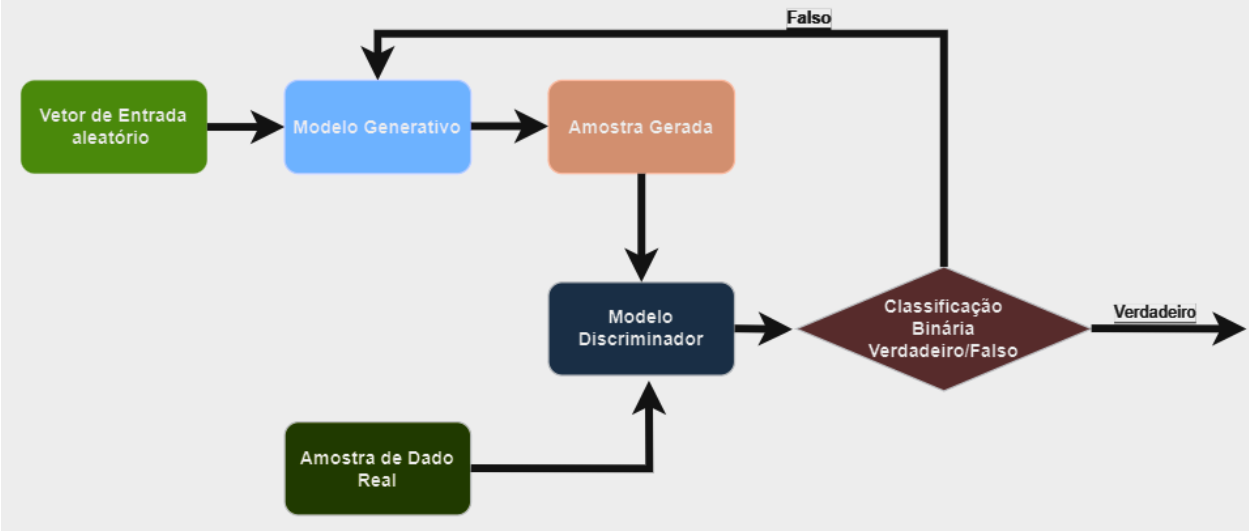
O sistema proposto será desenvolvido tendo como foco a detecção do rosto da imagem analisada, mas também a capacidade de identificar se o rosto detectado foi gerado por uma Inteligência Artificial do tipo Redes Adversárias Generativas (GAN). Para entender a sua definição é necessário primeiro apresentar o conceito de Rede Neural Generativa (RNG).

Uma Rede Neural Generativa (RNG) é capaz de gerar um número infinito de saídas tendo como base um conjunto finito de regras, sendo capaz de gerar um número infinito sons, textos e principalmente imagens, tantas possibilidades se devem ao fato da sua proposta ser estendida para qualquer padrão de entrada. Porém, o desafio real foi que essa geração de dados sintéticos não seja facilmente distinguível dos padrões reais, e também a possibilidade de serem mais controláveis. Essas características foram adquiridas com a chegada das Redes Adversárias Generativas.

Esse modelo proposto foi um grande salto na tecnologia referente a modelos generativos em 2014 por Goodfellow *et al.* (2014), as Redes Adversárias Generativas, mais conhecidas como GAN (Generative Adversarial Network), tornaram-se um dos exemplos mais notáveis da capacidade generativa da IA. O seu funcionamento consiste em um modelo generativo sendo confrontado com um adversário, um modelo discriminativo, que aprende a distinguir se uma amostra é proveniente da distribuição do modelo ou da distribuição dos dados.

Uma comparação feita por Goodfellow *et al.* (2014) foi de que o modelo generativo é similar a uma equipe de falsificadores, tentando produzir dinheiro falso e usá-lo sem ser detectado, enquanto o modelo discriminativo é comparável à polícia, tentando detectar o dinheiro falso. A competição nesse jogo impulsiona ambas as equipes a aprimorarem seus métodos até que as falsificações se tornem indistinguíveis dos artigos genuínos. A arquitetura desse modelo é representada na **Figura** 8 abaixo.

Figura 8 - Modelo da Arquitetura Geral de uma GAN



Fonte: Porkodi *et al.* (2023).

Com essa capacidade GANs tornaram-se extremamente úteis para diversas áreas, podendo ser usados para: Geração de Imagens com qualidade aprimorada, Super Resolução de Imagem, Pintura de Imagens, Reconhecimento de objetos, geração e predição de vídeos, transformação imagem/imagem, transformação texto/imagem, estimativa de pose humana, desobstrução de imagens e mineração de dados.

Segundo Porkodi *et al.* (2023) as GANs podem ser consideradas um dos avanços mais notáveis em aprendizado de máquina nos últimos anos, nos próximos anos é possível que elas sejam utilizadas em muitos mais domínios, incluindo a produção de infográficos a partir de texto, a criação de designs de sites, a compressão de dados, a descoberta e desenvolvimento de novos medicamentos, a geração de texto e música, entre várias outras aplicações.

Em campos nos quais a visão computacional desempenha um papel significativo, como fotografia, edição de imagens e jogos, entre outros, as GANs são empregadas porque aprendem a detectar e diferenciar imagens.

2.8 TENSORFLOW

O *TensorFlow* é um sistema de aprendizado de máquina que opera em larga escala em variados ambientes. Ele utiliza gráficos de fluxo de dados para representar computação, estados compartilhados e operações que alteram esse estado.

O TensorFlow mapeia os nodos do fluxo de dados de diversas máquinas em um cluster e dentro da máquina através de vários dispositivos computacionais, incluindo CPUs multiprocessados, GPUs de propósito geral e TPUs (Tensor Processing Units) que são ASICs personalizados. Sua versatilidade permite seu uso em diversas aplicações, com foco em treinamento de máquina e dedução de redes neurais. Muitos serviços da Google utilizam TensorFlow em produção, e sua utilização se tornou amplamente difundida em pesquisas de aprendizado de máquina (FALCÃO *et al.*, 2019).

Um dos campos em que o TensorFlow desempenha um papel crucial é o reconhecimento e classificação de imagens. Os algoritmos baseados em Redes Neurais Convolucionais (CNNs) implementados no TensorFlow têm se destacado nessa área. Essas redes são especialmente eficazes para tarefas de classificação de imagens, como identificação de objetos, detecção de padrões e segmentação de conteúdo visual. A capacidade de aprender com experiência e a flexibilidade para adaptar-se a diferentes domínios tornam o TensorFlow uma ferramenta essencial para enfrentar desafios do mundo real, como diagnóstico médico, veículos autônomos e análise de imagens em geral.

2.9 KERAS

A tf.keras, a API de alto nível do TensorFlow para criação e treinamento de modelos de aprendizado profundo, destaca-se por sua facilidade de uso, modularidade e capacidade de extensão. Com uma interface simples e consistente, otimizada para os casos de uso comuns, ela oferece um feedback claro e prático para os erros do usuário, tornando a prototipagem rápida, a pesquisa de ponta e a produção mais acessíveis.

Além disso, os modelos modulares e compostos da Keras são construídos conectando elementos configuráveis, proporcionando flexibilidade e poucas restrições na criação de arquiteturas complexas. A facilidade de extensão é outra vantagem crucial, permitindo o desenvolvimento de elementos personalizados que expressam novas ideias para pesquisa, incluindo a criação de novas camadas, métricas e funções de perda para impulsionar o desenvolvimento de modelos de última geração (TENSORFLOW, 2020).

2.10 GOOGLE COLAB

Google Colab, ou Colaboratory, é uma plataforma gratuita baseada na nuvem oferecida pelo Google. Ele fornece um ambiente de notebook interativo e colaborativo que permite a criação e execução de código diretamente no navegador, sem a necessidade de configurar ou instalar qualquer software no computador (GUNAWAN *et al.*, 2020). com ele é possível usar os recursos do Python e as suas bibliotecas sem a necessidade de configurar um ambiente de desenvolvimento local.

3 Trabalhos Relacionados

Esse trecho tem como foco apresentar trabalhos relacionados as tecnologias e métodos utilizados neste projeto, para que sirvam de base para uma análise e comparação do foco do trabalho, abordagem, metodologias e aplicação.

3.1 Olhos dizem tudo: Pupilas Irregulares revelam faces irregulares geradas por GAN

Esse estudo foi realizado por Guo *et al.* (2022), pesquisadores de Singapura, em que propuseram um novo método de detecção de rosto gerados por GAN, isso foi motivado pela simples observação de que as faces geradas exibiam a comum característica de formas das pupilas irregulares. As pupilas especificamente humanas devem aparecer como suaves círculos ou elipses, mas em contraste as pupilas dos rostos gerados por GAN apresentam contornos e formas irregulares.

Isso se deve principalmente à falta de compreensão da anatomia do olho humano em questão de forma e geometria. Portanto, o método de GUO *et al.* (2022) consiste em localizar os olhos para então segmentar a região da pupila, é usado um modelo elíptico para ser ajustado de forma paramétrica ao contorno da pupila, isso é usado para o cálculo de circularidade das pupilas como mostrado na **Figura** 9 abaixo.

Figura 9 - Pupilas de uma pessoa real na esquerda e GAN na direita, é importante notar a forte forma circular nas pupilas dos olhos reais (amarelo) em comparação com as pupilas dos olhos direitos (vermelhos).

Fonte: Guo *et al.* (2022)

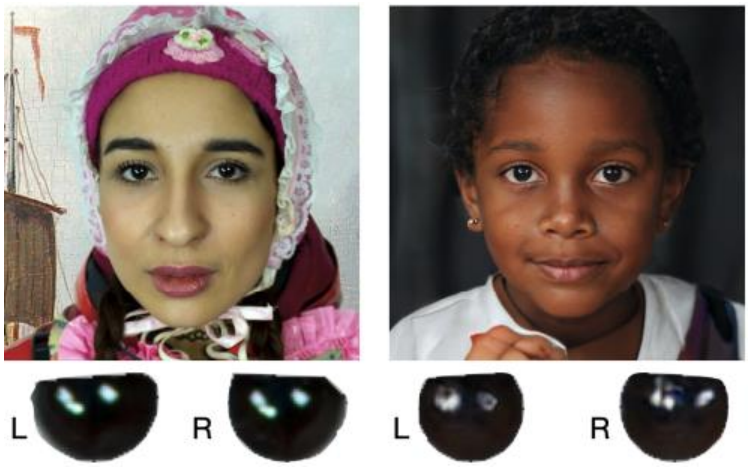
3.2 Expondo Rostos Gerados por GAN Usando Destaques Especulares da Córnea Inconsistentes

Os estudos apresentados por Hu *et al.* (2021) mostram um método que consiste em analisar inconsistências entre os olhos de uma face gerada por GAN. Mais especificamente, são detectados e alinhados os destaques especulares das córneas dos olhos, permitindo uma comparação pixel a pixel para medir o Intersecção de União (IoU).

A premissa é que olhos humanos reais, capturados por uma câmera na configuração de retrato, devem exibir uma grande semelhança entre os destaques da córnea em ambos os olhos. Porém, essa suposição não se aplica aos olhos gerados por GAN, nos quais podem ocorrer certas inconsistências, como números diferentes, formas geométricas diversas ou localizações relativas distintas das corneais.

Deve-se ressaltar que esse método opera melhor com base em certas condições, como a pose frontal em retrato, fonte(s) de iluminação distante(s) e a presença dos destaques nos olhos, e a falta dessas condições pode levar a um aumento significativo de falsos positivos. Em resumo, esses métodos de detecção, fundamentados em princípios físicos, demonstram ser mais resilientes contra ataques adversários, proporcionando resultados interpretáveis de maneira intuitiva para os usuários humanos (Hu *et al.*, 2021).

Figura 10 – Método de detecção através do reflexo córneo.



Fonte: Hu *et al.* (2021).

Na **Figura** 10, em sua parte superior: Destaques da córnea para um rosto humano real (esquerda) e um rosto gerado por GAN (direita). Inferior: Às regiões das córneas estão isoladas e dimensionadas para melhor visibilidade. É possível observar que o reflexo nos olhos do rosto esquerdo é similar enquanto que os do rosto do lado direto gerado pela GAN são diferentes

3.3 Detecção de deepfakes em tempo real em videoconferências com sondagem ativa e reflexo corneano

Em um trabalho realizado por Guo, Wang e Lyu (2023), pesquisadores da Universidade de Buffalo, Estado de Nova Iorque, EUA. O projeto tinha como objetivo a criação de um novo método forense para fazer a detecção de deepfakes em tempo real durante vídeo conferências.

Figura 11 - Exemplos de deepfakes usados em vídeo conferência, para cada par: esquerda: faces usadas como modelos, direita: deepfakes.

Fonte: GUO, WANG & LYU (2023).

O método consistia em exibir um padrão distinto na tela durante uma vídeos chamada e usar o reflexo corneano extraído das imagens do rosto do participante da chamada. Esse padrão pode ser induzido por um participante da chamada exibindo em uma tela compartilhada ou integrado diretamente ao cliente de videochamada. Em ambos os casos, nenhum hardware especializado de imagem ou iluminação foi necessário, esse método está sendo melhor ilustrado na **Figura 12** abaixo.

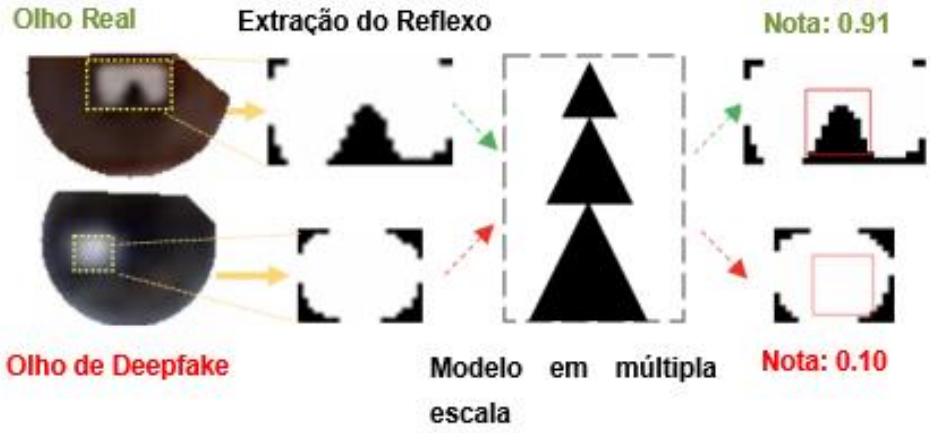
Figura 12 - Demonstração do Método

Fonte: Guo, Wang & Lyu (2023).

Na **Figura 12** mostrada uma participante de uma videochamada sendo autenticada ativamente com os padrões ao vivo exibidos na tela. Direita: A córnea de uma pessoa real produzirá uma imagem do padrão mostrado na tela, enquanto um DeepFake em tempo real não conseguirá.

O método de detecção consiste em localizar a região dos olhos a partir dos pontos faciais, em seguida a parte da íris é segmentada usando um detector de bordas e transformada de Hough, então essas imagens segmentadas das írises são submetidas a etapas de correspondência de modelo para a detecção, o padrão da imagem é comparada com o reflexo corneano. A compatibilidade entre os dois indica uma pessoa real, enquanto que a incompatibilidade indica uma possível imitação com *Deepfake* em tempo real. O método de verificação é ilustrado na **Figura 13** a seguir.

Figura 13 - Método de reconhecimento pelo reflexo corneano



Fonte: Guo, Wang & Lyu (2023).

Foi avaliada a confiabilidade dessa abordagem por meio de simulações em larga escala, as simulações foram feitas com dois conjuntos de dados, sendo um com vídeo conferências com participantes reais e suas sínteses deepfakes. E a outra forma foi com a gravação de dois usuários a 30 cm da tela e câmera em diferentes ambientes. Como resultado foi descoberto que a iluminação interna tem uma influência limitada no desempenho do método, pois o alto contraste diminui a importância da luz no ambiente.

3.4 Comparativo entre os projetos

A Tabela 1 faz um comparativo dos trabalhos de acordo com o Foco e a Abordagem dos projetos.

Tabela 1: Foco e Abordagem dos projetos

Parâmetros	GUO *et al.* (2022)	HU *et al.* (2021)	GUO, WANG & LYU (2023)	Projeto Proposto
Foco	**Propor Método para detecção**	**Criação de método de detecção**	**Segurança**	**Prevenção e Segurança**
Método	**Detecção e Classificação de Pupilas**	**Reflexos da Córnea**	**Sondagem Ativa e Reflexo Corneano**	**Aprendizado de Máquina**
Abordagem	**Experimental e Aplicação**	**Experimental**	**Aplicação**	**Experimental e Simulação**

Fonte: O autor (2024).

Todos os projetos tem como foco a detecção de faces geradas através dos olhos, sendo pelo formato da íris, ou pelos reflexos corneais, de fato demonstraram serem ótimas de identificar fotos gerados, por outro lado esses tipos de técnicas não são viáveis para fotos em que o rosto gerado tem óculos, há também questões como doenças e infecções nos olhos que podem afetar o reconhecimento tal como citado por Guo *et al.* (2022).

4 Desenvolvimento

Nesse tópico serão explicadas as etapas feitas para o desenvolvimento do projeto, passando pelas etapas de:

1. Preparação do Dataset
2. Preparação do Ambiente
3. Formatação das imagens
4. Pré-processamento das imagens
5. Criação do Modelo da Arquitetura
6. Treinamento da Inteligência Artificial

O desenvolvimento do sistema de Classificação de Faces Verdadeiras e Falsas foi feito pelo Google Colab, que utiliza a linguagem Python.

4.1 Preparação do dataset

O sistema conta com dois conjuntos que precisam ser classificados, são as classes “real” e “fake”. Por isso é essencial que haja um conjunto de imagens para o treinamento, correção e teste para cada classe. O conjunto para a classe “real” precisa ser constituído por imagens de pessoas reais, enquanto que o conjunto para a classe “fake” precisa ser constituído por imagens de rostos gerados por GAN.

O conjunto de imagens usadas no projeto está disponível no link https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces, esse banco de dados é composto por 140 mil imagens, metade são de rostos de pessoas reais e a outra metade são de faces geradas por inteligência artificial, esse diretório é dividido em 3 pastas principais: treino, validação e teste. Cada pasta tem duas subpastas para separar as imagens reais das imagens fakes.

As técnicas de Aprendizado de Máquina são centradas nos dados, o que implica que elas são capazes de aprender de forma automática a partir de conjuntos extensos de informações. No contexto do projeto essas informações são as imagens e seus rótulos. A partir dos dados lidos o algoritmo deverá ser capaz de criar suposições sem precisar de instruções adicionais.

A inferência indutiva é um dos principais métodos empregados para extrair novo conhecimento e prever eventos futuros no Aprendizado de Máquina. No entanto, a generalização pode não ser realizada de forma precisa na inferência indutiva. A qualidade dos dados tem total influência na precisão das generalizações, portanto, os dados mais precisos resultam em generalizações mais precisas também no contexto do Aprendizado de Máquina (LUDERMIR, 2021).

Figura 14 – Exemplo de imagens usadas no treinamento do modelo

Fonte: Xhlulu (2020).

Em resumo, isso significa que dados precisos geram generalizações mais precisas, enquanto que a presença de dados inadequados gera resultados insatisfatórios, fazendo com que o funcionamento da Inteligência Artificial esteja fora do ideal. Esse é o motivo de não ser necessário o uso de todas as imagens disponíveis no banco de dados, para testar a importância da quantidade de dados na capacidade de classificação serão feitos dois sistemas, ambos com quantidades diferentes de imagens do mesmo dataset.

A quantidade de imagens e os seus devidos agrupamentos está sendo mostrada na **Tabela 2.**

Tabela 2: Distribuição de Dados

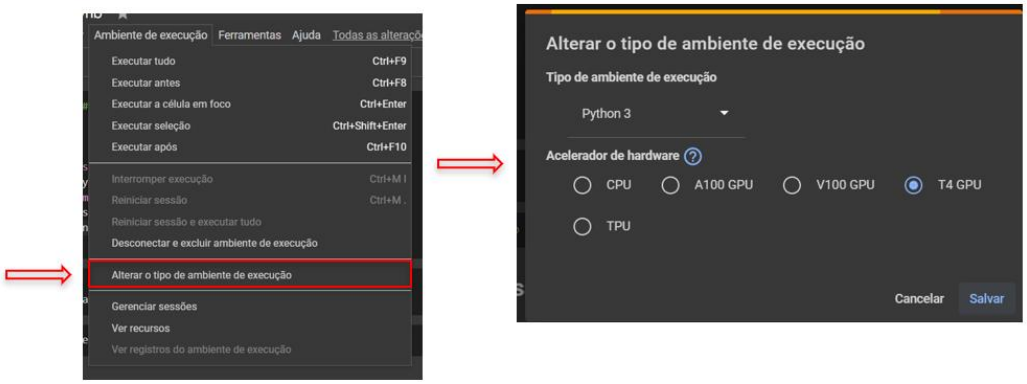
	Train		Valid		Test		Total	
Classes	**Sistema 1**	**Sistema 2**	**Sistema 1**	**Sistema 2**	**Sistema 1**	**Sistema 2**	**Sistema 1**	**Sistema 2**
Real	1612	3970	1488	1980	1004	1981	4104	7931
Fake	1603	3983	1603	1998	1003	2002	4209	7983

Fonte: O autor (2024).

4.2 Preparação do Ambiente

Um dos passos mais importantes para o desenvolvimento do projeto é a configuração do ambiente. A facilidade de configurar o ambiente de programação no Google Colab. foi o principal motivo para a sua escolha, a forma de escolher uma TPU para a execução do projeto é simples, como mostrado na **Figura 15** a seguir.

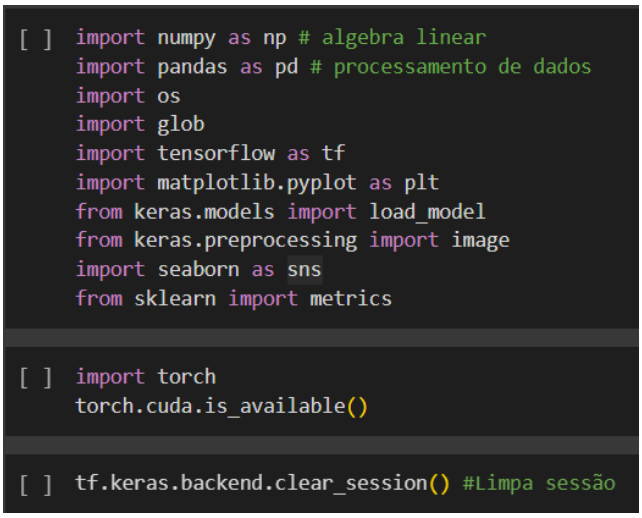
Figura 15 - Tutorial de como habilitar a execução do ambiente com TPU.



Fonte: O autor (2024).

Com a TPU habilitada, o pré-processamento dos dados e o treinamento da Rede Neural serão mais rápidos e não exigirão muito poder computacional de uma máquina pessoal. Portanto o próximo passo são as importações necessárias para o desenvolvimento do projeto da forma mostrada na **Figura 16**.

Figura 16 - Importações de Bibliotecas essenciais para o desenvolvimento do projeto



Fonte: O autor (2024).

Essas bibliotecas são úteis para trabalhar com dados e modelos de aprendizagem de máquina.

- A biblioteca “*numpy*” serve para dar suporte a matrizes grandes e arrays multidimensionais.
- A biblioteca “*pandas*” oferece uma estrutura de dados e operações para manipulação das tabelas numéricas.
- As bibliotecas *“os”* e “*glob”* permitem a interação com o sistema operacional, no caso auxiliam no acesso aos arquivos do *dataset*.
- A biblioteca “*tensorflow”* é a principal biblioteca do projeto por auxiliar na construção da Aprendizagem de Máquina e Redes Neurais Artificiais.
- A “*matplotlib.pyplot”* são uma coleção de funções que a permitem que o *matplot* funcione como o MATLAB.

- “*keras”* estão sendo exportadas funções que ajudam a carregar o modelo salvo e que fazem o pré-processamento de imagens.
- O “*seaborn”* é para a visualização dos dados, permitindo o desenho de gráficos estatísticos.
- Do “*sklearn*” está sendo importada a função *metrics* para a avaliação do desempenho.

Para verificar se o ambiente está de fato usando uma TPU ou GPU é necessário listar os dispositivos físicos usados como mostrado na **Figura 17**:

Figura 17 - Algoritmo para verificação de disponibilidade de GPU/TPU

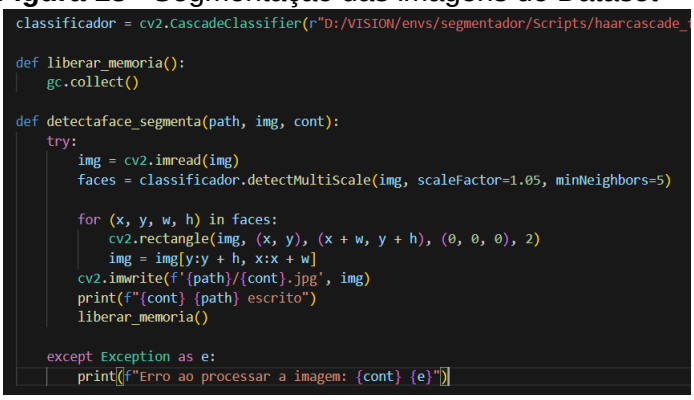
```
gpus = tf.config.list_physical_devices("GPU")
print(gpus)
if gpus:
    for gpu in gpus:
        print("GPU encontrada")
    else:
        print("Falha na detecção da GPU")
```

Fonte: O autor (2024).

4.3 Formatação das imagens

Para melhor assegurar os dados para o treinamento foi usada uma função, mostrada na imagem **Figura 18,** que recebia os endereços das imagens a partir de uma estrutura de repetição, tentava ler e reconhecer alguma face, se reconhecida a região de interesse da face é segmentada e salva em uma outra pasta seguindo a mesma estrutura do dataset original.

Figura 18 - Segmentação das imagens do Dataset



```
classificador = cv2.CascadeClassifier(r"D:/VISION/envs/segmentador/Scripts/haarcascade_

def liberar_memoria():
    gc.collect()

def detectaface_segmenta(path, img, cont):
    try:
        img = cv2.imread(img)
        faces = classificador.detectMultiScale(img, scaleFactor=1.05, minNeighbors=5)

        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 0), 2)
            img = img[y:y + h, x:x + w]
        cv2.imwrite(f'{path}/{cont}.jpg', img)
        print(f"{cont} {path} escrito")
        liberar_memoria()

    except Exception as e:
        print(f"Erro ao processar a imagem: {cont} {e}")
```

Fonte: O autor (2024).

Com os arquivos compactados e armazenados na nuvem através do Google Drive é necessária a sua extração.

Figura 19 - Extração do dataset

```
from google.colab import drive
drive.mount('/content/gdrive')
!unzip gdrive/My\ Drive/data/midarchive.zip > /dev/null
```

Fonte: O autor (2024).

Depois da extração do conjunto de imagem, é necessário listar todos os arquivos das subastas, isso também serve para verificar todos os dados estão corretamente organizados.

Figura 20 – Acesso aos dados do projeto

```
PATH = "/content/midarchive/real_vs_fake"

train_dir = os.path.join(PATH,"train")
test_dir = os.path.join(PATH,"test")
valid_dir = os.path.join(PATH,"valid")
print("Subpastas em Train: ",os.listdir(train_dir))
print("Subpastas em Teste: ",os.listdir(test_dir))
print("Subpastas em Valid: ",os.listdir(valid_dir))
```

Fonte: O autor (2024).

Como mostrado **Figura** 20 acima, as variáveis *train_dir, valid_dir e test_dir* armazenam os caminhos para as suas respectivas pastas. Depois da verificação das subpastas do projeto é criado um DataFrame para que haja um padrão organizacional, tendo como base as informações das imagens como a pasta em que estão organizadas seu caminho completo e o rótulo associado, sendo ele “real” ou “fake”, ao executar o código da **Figura** 21 é gerada a saída mostrada na **Figura 22**. A organização foi feita com um DataFrame de 3 colunas e o número de linhas corresponde a legenda mais o número de imagens sendo usadas no total.

Figura 21 - Criação de um DataFrame

```
images_df = {
    "folder":[],
    "image_path":[],
    "label":[]
}

for folder in os.listdir(PATH): #intera sobre cada subpasta
    for label in os.listdir(PATH + "/" + folder): #intera sobre as pastas real e fake (colunas)
        for img in glob.glob(PATH + "/" + folder + "/" + label + "/*.jpg"):
            images_df["folder"].append(folder)
            images_df["image_path"].append(img)
            images_df["label"].append(label)

images_df = pd.DataFrame(images_df)
images_df
```

Fonte: O autor (2024).

Figura 22 - DataFrame gerado

	folder	image_path	label
0	segvalid	/content/seg_midarchive/segvalid/fake/1508.jpg	fake
1	segvalid	/content/seg_midarchive/segvalid/fake/1560.jpg	fake
2	segvalid	/content/seg_midarchive/segvalid/fake/402.jpg	fake
3	segvalid	/content/seg_midarchive/segvalid/fake/1165.jpg	fake
4	segvalid	/content/seg_midarchive/segvalid/fake/1114.jpg	fake
...	...	...	...
15909	segtrain	/content/seg_midarchive/segtrain/real/3011.jpg	real
15910	segtrain	/content/seg_midarchive/segtrain/real/993.jpg	real
15911	segtrain	/content/seg_midarchive/segtrain/real/1041.jpg	real
15912	segtrain	/content/seg_midarchive/segtrain/real/1163.jpg	real
15913	segtrain	/content/seg_midarchive/segtrain/real/3690.jpg	real

15914 rows × 3 columns

Fonte: O autor (2024).

4.4 Pré-processamento das imagens

Um procedimento fundamental no treinamento de Redes Neurais é o pré-processamento das imagens, nessa etapa ocorre a normalização que é o redimensionamento dos pixels para uma escala específica, o que acelera a convergência da Rede Neural durante o treinamento. Adicionalmente, a aplicação de transformações como rotação, flip e zoom, para redução de variância, torna a rede insensível a tais mudanças, isso garante uma classificação precisa das imagens, que independente da sua orientação ou escala a rede deverá ser capaz de classificar de forma precisa.

Essas técnicas não só aprimoram o desempenho do modelo, mas também otimizam o uso de memória, especialmente ao redimensionar imagens para dimensões menores, agilizando o processo de treinamento. Aplicar transformações aleatórias às imagens (como flip horizontal) aumentam o tamanho do conjunto de dados de treinamento. Isso pode ajudar a melhorar o desempenho do modelo, especialmente se o conjunto de dados original for pequeno. Com as informações das imagens já formatadas é possível seguir para a parte de pré-processamento como mostrado na **Figura 23**.

Figura 23 - Pré-Processamento das imagens

```
image_train_gen = tf.keras.preprocessing.image.ImageDataGenerator(
                                    rescale=1./255.,
                                    horizontal_flip=True,
                                    )

image_gen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255.)

train_ds = image_train_gen.flow_from_directory(
    train_dir,
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
)

valid_ds = image_gen.flow_from_directory(
    valid_dir,
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary'
)

test_ds = image_gen.flow_from_directory(
    test_dir,
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    shuffle=False) #reprodução aleatória desativada
```

Fonte: O autor (2024).

- *“image_train_gen”* é um gerador de dados de imagem que redimensiona as imagens para uma escala de 0 a 1 e aplica um flip horizontal aleatório.
- *“image_gen*” é um gerador de dados de imagem que apenas redimensiona as imagens para uma escala de 0 a 1.
- *“train_ds”, “valid_ds”* e “*test_ds”* são conjuntos de dados de treinamento, validação e teste, respectivamente. Eles são criados usando o método “*flow_from_directory”* que carrega imagens de um diretório, redimensiona para (256, 256), somente no “*train_ds”* o flip horinzontal aleatório é aplicado, em todos o *class_mode = ‘binary’* está presente, ele significa que é uma tarefa de classificação binária.
- Para o conjunto de dados de teste “*test_ds*”, a opção de embaralhamento está desativada “*shuffle=False*”. Isso é útil pois mantém a ordem original das imagens para correspondência posterior com as previsões.

O resultado do pré-processamento pode ser visto ao executar o código da **Figura 24** a sua saída pode ser verificada na **Figura 25**.

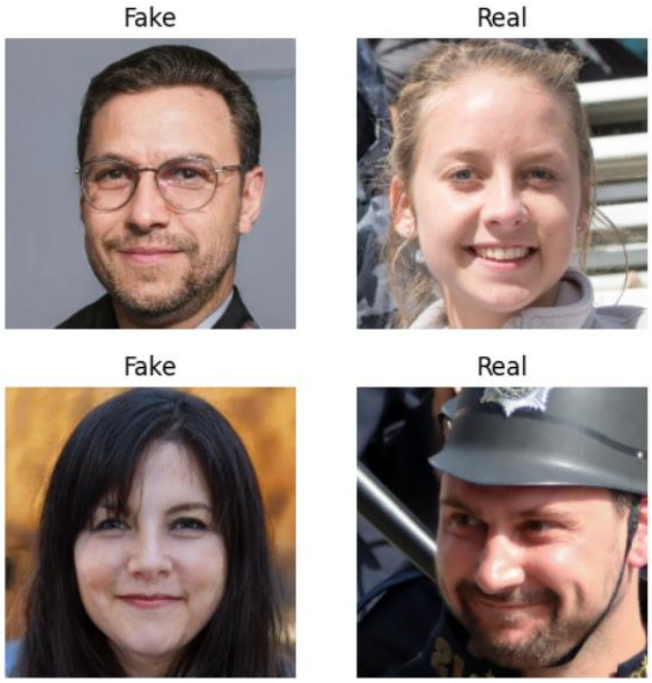
Figura 24 - Código para a verificação de imagens pré-

```
def plot_images(img, label):
    plt.figure(figsize=[12, 12])
    for i in range(16):
        plt.subplot(4, 4, i+1)
        plt.imshow(img[i])
        plt.axis('off')
        if label[i] == 0:
            plt.title("Fake")
        else:
            plt.title("Real")

img,lbl = next(train_ds) #Somente imagens da pasta de treino
plot_images(img,lbl)
```

Fonte: O autor (2024).

Figura 25 - Saída de imagens pré-processadas



Fonte: O autor (2024).

4.5 Criação Do Modelo da Arquitetura

Depois do pré-processamento é feito, pode-se prosseguir para a criação do modelo da arquitetura, para esse projeto foi escolhida a DenseNet121, trata-se de uma Rede Neural Convolucional pré-treinada no conjunto de dados *ImageNet*, ela é uma variação da família *DenseNet*, ela foi apresentada pela primeira vez no trabalho de Huang *et al.* (2016), a sua proposta foi lidar com o desafio de Redes Neurais Convolucionais Profundas, em busca de melhorar a eficiência de treinamento e a precisão na classificação de imagens.

O diferencial da arquitetura *DenseNet* é que ela conecta cada camada a todas as outras, isso promove uma propagação de características mais eficaz e mitiga o problema de gradientes desaparecidos. O fato de ter uma conectividade densa também facilita a reutilização de características aprendidas, no resulta em uma redução significativa no número de parâmetros necessários. Essa abordagem se traduz em melhorias substanciais no desempenho em benchmarks como CIFAR-10, CIFAR-100, SVHN e

ImageNet, com menor carga computacional, tornando o *DenseNet* uma escolha atraente em aplicações de visão computacional.

Desde então essa rede pôde ser usada em diversos trabalhos como o de Zare e Pourkazemi (2021) para fazer a segmentação e classificação de imagens de lesões dermatoscópicas na pele, fator muito útil para detecção precoce de câncer. Também na área da saúde houve o trabalho de Zhang *et al.* (2021) que consistia em detectar com 80% de precisão a Retinopatia Diabética que é um tipo de cegueira causada pela diabetes.

Na **Figura 26** é mostrado o código responsável pela geração da arquitetura do modelo, a função "*get_model()"* recebe como argumento a tupla "*input_shape"* com os valores das imagens pré-processadas, que são 256x256 pixels com 3 canais de cores. A saída da *DenseNet* é carregada por uma camada de GlobalAveragePooling2d, que é responsável por calcular a média dos valores em cada mapa de características. Em seguida, a saída é passada por uma camada densa com 512 neurônios e função de ativação ReLU.

A camada *BatchNormalization* é usada para normalizar a ativação da camada anterior, o que pode acelerar o treinamento e melhorar o desempenho do modelo. A camada *Dropout* com taxa de 0.3 é usada para prevenir o *overfitting*. Ela aleatoriamente define uma fração de 0.3 das entradas para zero durante o treinamento. Finalmente, a camada de saída é uma camada densa com 1 neurônio e função de ativação *sigmoid*, que é adequada para classificação binária.

O modelo é compilado com a função de perda *binary_crossentropy* por ser adequada para classificação binária, o otimizador *adam*, e a métrica de avaliação é pela acurácia.

Figura 26 - Função que gera a arquitetura do modelo

```
input_shape = (256, 256, 3) #define a forma da entrada dos dados

#cria o modelo da arquitetura e compila
def get_model(input_shape):

    input = tf.keras.Input(shape=input_shape)

    densenet = tf.keras.applications.DenseNet121( weights="imagenet",
                                                   include_top=False,
                                                   input_tensor = input)

    x = tf.keras.layers.GlobalAveragePooling2D()(densenet.output)
    x = tf.keras.layers.Dense(512, activation='relu')(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    output = tf.keras.layers.Dense(1, activation='sigmoid')(x) #classificação binária

    model = tf.keras.Model(densenet.input, output)

    model.compile(loss="binary_crossentropy",
                  optimizer="adam", metrics=["accuracy"])

    return model
model_ft = get_model(input_shape)
```

Fonte: O autor (2024).

4.6 Treinamento da Inteligência Artificial

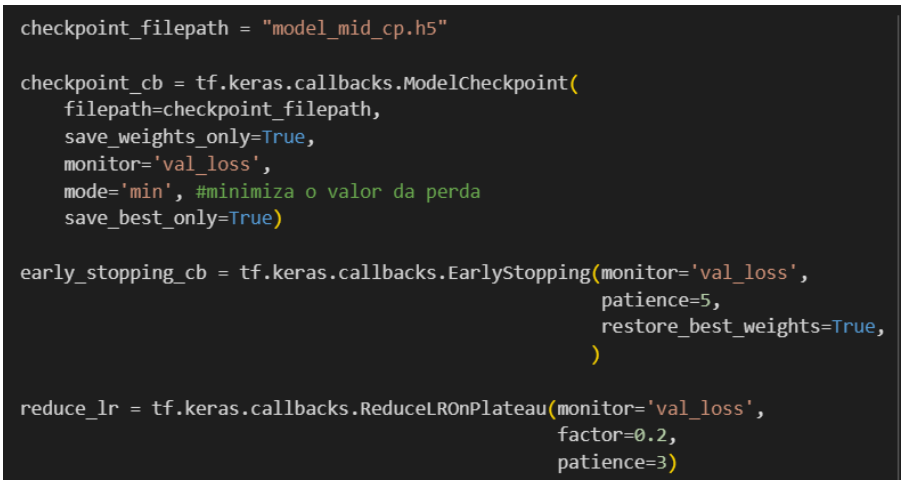
Finalmente na etapa de treinamento da IA é necessário estabelecer os *callbacks* do TensorFlow para o treinamento de um modelo, eles têm como função avaliar o comportamento de um modelo *Keras* durante o treinamento.

- O callback *ModelCheckpoint* é responsável por salvar os pesos do modelo em um arquivo específico sempre que a perda de validação atinge um novo mínimo, permitindo retomar o treinamento de onde parou.

- O *EarlyStopping* interrompe o treinamento se a perda de validação não melhorar após um número definido de épocas, isso evita o *overfitting* e restaura os pesos do modelo para o melhor estado.
- O *ReduceLROnPlateau* reduz a taxa de aprendizado quando a perda de validação estagna, auxiliando o modelo a sair de mínimos locais.

Conforme mostrado na **Figura 27** esses *callbacks* serão usados para monitorar e ajustar o processo de treinamento.

Figura 27 - Definição dos *Callbacks*



```
checkpoint_filepath = "model_mid_cp.h5"

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_loss',
    mode='min', #minimiza o valor da perda
    save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                     patience=5,
                                                     restore_best_weights=True,
                                                    )

reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                 factor=0.2,
                                                 patience=3)
```

Fonte: O autor (2024).

Depois de estabelecidos os *callbacks* já é possível começar o treinamento da Rede Neural, conforme mostrado na Erro! Fonte de referência não encontrada.**28** é usada a função *".fit()"* do *tensorflow* isso serve para ajustar o modelo aos dados do treinamento, que no caso estão na variável "*train_ds*", como o *batch_size* foi definido anteriormente como 32, então para uma atualização dos pesos (*backpropagation*) serão usadas 32 imagens por vez, o trecho do *dataset* composto por imagens de treino para os grupos real e fake tem 7953 imagens no total serão utilizados 249 *backpropagations* por época (7953/32 ≈ 248,53).

Um dos parâmetros mais importantes é a definição de épocas que foram definidas para serem 15, se o modelo tiver poucas épocas ele poderá não ser capaz de aprender de forma adequada os padrões dos dados de treino sendo isso a causa do *underfitting*,

isso resulta na incapacidade do modelo para fazer uma boa classificação com os dados de teste, por outro lado, se o número de épocas for muito grande o *overfitting* poderá ser a consequência, devido a isso o número de épocas deve ser equilibrado (“ QUAL É A IMPORTÂNCIA DE TREINAR...”, 2023). Os dados de validação correspondem as imagens pré-processadas da pasta de validação, em cada época o modelo será avaliado nesse conjunto de dados de validação.

Outros parâmetros são os *callbacks* que precisam ser postos em formato de lista. O histórico de treinamento de todo o processo está sendo armazenado em *“history_ft”*.

Figura 28 - Treinamento do Modelo com a função .fit() do Tensorflow

```
history_ft = model_ft.fit(train_ds,
                          epochs = 15,
                          validation_data = valid_ds,
                          callbacks=[checkpoint_cb, early_stopping_cb, reduce_lr])
```

```
Epoch 1/15
249/249 [==============================] - 196s 519ms/step - loss: 0.6444 - accu
Epoch 2/15
249/249 [==============================] - 120s 484ms/step - loss: 0.4342 - accu
Epoch 3/15
249/249 [==============================] - 127s 508ms/step - loss: 0.3330 - accu
Epoch 4/15
249/249 [==============================] - 127s 511ms/step - loss: 0.2599 - accu
Epoch 5/15
249/249 [==============================] - 120s 483ms/step - loss: 0.1916 - accu
Epoch 6/15
249/249 [==============================] - 120s 483ms/step - loss: 0.1676 - accu
Epoch 7/15
249/249 [==============================] - 126s 508ms/step - loss: 0.1495 - accu
Epoch 8/15
249/249 [==============================] - 121s 487ms/step - loss: 0.0662 - accu
Epoch 9/15
249/249 [==============================] - 127s 508ms/step - loss: 0.0362 - accu
Epoch 10/15
249/249 [==============================] - 121s 484ms/step - loss: 0.0317 - accu
Epoch 11/15
249/249 [==============================] - 120s 483ms/step - loss: 0.0327 - accu
Epoch 12/15
249/249 [==============================] - 127s 511ms/step - loss: 0.0189 - accu
Epoch 13/15
249/249 [==============================] - 121s 486ms/step - loss: 0.0133 - accu
```

Fonte: O autor (2024).

5 Resultados

Para a ilustração dos resultados obtidos pelos sistemas de classificação serão usados gráficos de desempenho e matrizes de confusão para a parte de Resultados Quantitativos, na parte dos Resultados Qualitativos serão mostradas as imagens do dataset separadas para a fase de teste. As explicações sobre as ilustrações gráficas servem para ambos os sistemas, a diferença entre eles está nos valores.

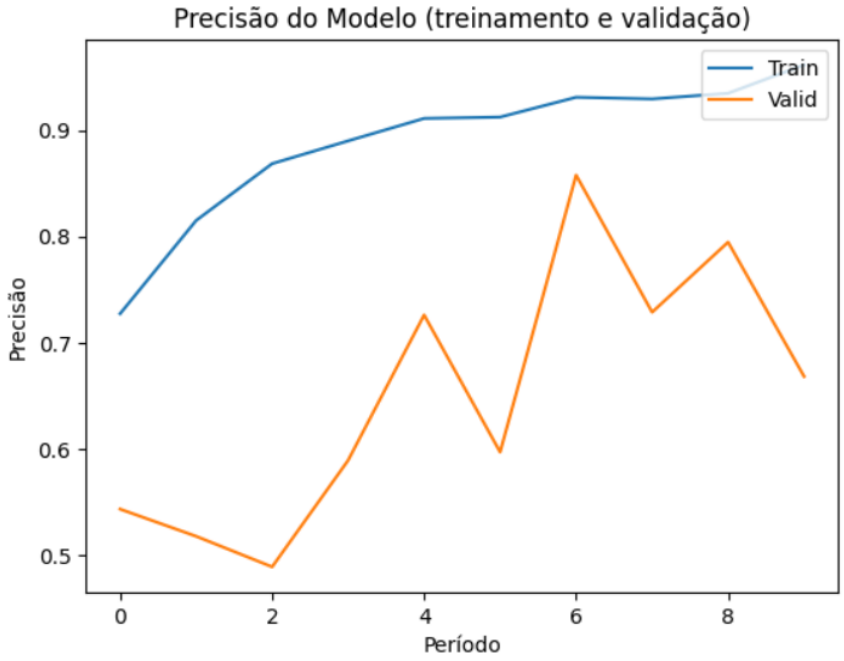
5.1 Resultados do Sistema 1

Aqui serão apresentados os resultados do Sistema 1, nele foram usadas 3215 imagens para o treino do modelo, 3091 imagens para a validação e 2007 para os testes, o *bach_size* foi igual a 64, portanto, o modelo precisou de 51 *backpropagations* nas 10 épocas.

5.1.1 Resultados Quantitativos

O gráfico da **Figura** 29 permite visualizar o desempenho do modelo do sistema 1 durante o treinamento.

Figura 29 - Gráfico da Precisão do Modelo do Sistema 1



Fonte: O autor (2024).

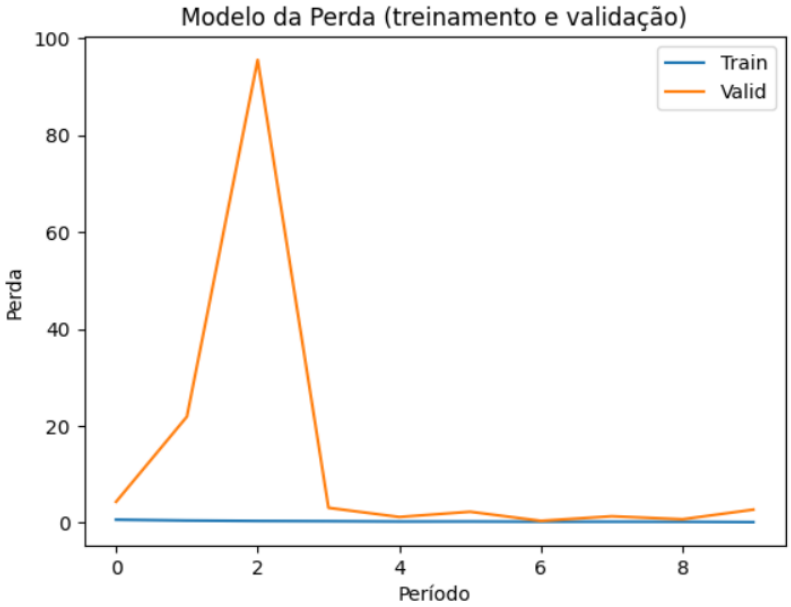
A seguir uma explicação que também servirá para o modelo de gráficos de desempenho do Sistema 2:

- **Eixo X (Período)**: Este eixo representa o número de períodos (ou épocas) durante o treinamento. Cada período é uma passagem completa através do conjunto de dados de treinamento.
- **Eixo Y (Precisão)**: Este eixo representa a precisão do modelo, que é a proporção de previsões corretas feitas pelo modelo. A precisão varia de 0 a 1, onde 1 significa que todas as previsões do modelo estão corretas.
- **Linha Azul (Train)**: Esta linha representa a precisão do modelo no conjunto de treinamento durante cada período. Você pode ver que a precisão do treinamento aumenta com o tempo, o que é esperado à medida que o modelo aprende com os dados de treinamento.
- **Linha Laranja (Valid)**: Esta linha representa a precisão do modelo no conjunto de validação durante cada período. A precisão de validação é uma medida do desempenho do modelo em dados novos e não vistos.

No gráfico é visto que a precisão do sistema oscila entre 0.5 e 1.0, a precisão do treinamento tem um crescimento gradual ao ponto de quase chegar em 1.0, porém, é perceptível que a precisão da validação oscila muito em diversos períodos.

As eventuais quedas podem ser um sinal de *overfitting*, que é quando o modelo captura ruído nos dados de treinamento, a queda indica que ele não generalizou bem para novos dados, mas de alguma forma conseguiu se recuperar. O *overfitting* ocorre quando o modelo aprende muito bem os dados de treinamento, incluindo o ruído e os detalhes específicos, que não se generalizam para novos dados.

Figura 30 - Gráfico de Perda do Modelo do Sistema 1



Fonte: O autor (2024).

A **Figura** 30 mostra o gráfico gerado para ilustrar a perda do modelo durante o treinamento, o aumento de perda na validação durante a época 2 e sua estabilização em seguida é um dos sinais do *overfitting* sob outra perspectiva. Outra forma de avaliar o modelo é usando a função "*evaluate()"* do *Tensorflow*, ela é uma ferramenta integrada que permite avaliar o desempenho de um modelo, ela faz isso usando os dados de teste e retorna as perdas e as métricas de avaliação especificada, essa verificação é mostrada na **Figura** 31.

Figura 31 – Avaliação de desempenho do Sistema 1

```
# Avalia o modelo
test_loss, test_acc = model.evaluate(test_ds)
print("Teste Perda:", test_loss)
print("Test Precisão:", test_acc)
```

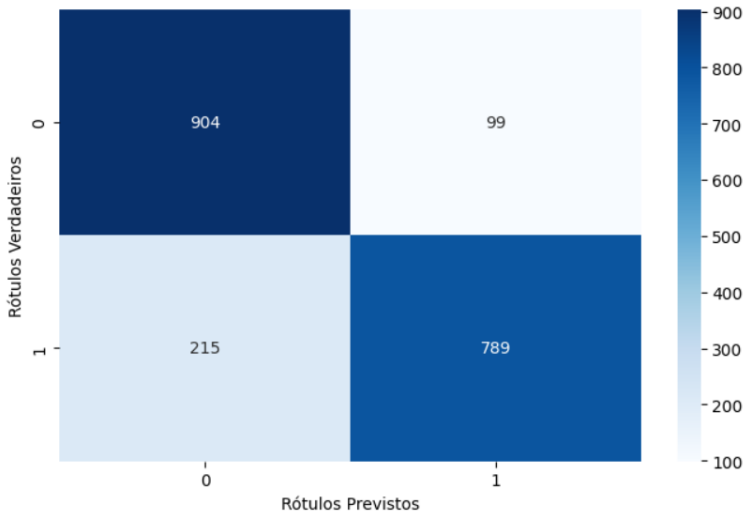
```
32/32 [==============================] - 10s 227ms/step - loss: 0.3829 - accuracy: 0.8435
Teste Perda: 0.3829296827316284
Test Precisão: 0.8435475826263428
```

Fonte: O autor (2024).

Como mostrado na **Figura** 31, mesmo com o problema do *overfitting* o sistema teve um nível de precisão de 84.35% para os dados de teste. Para casos de necessidade em melhorar ainda mais o modelo, a simples tarefa de carregar os pesos novamente pode melhorar essa porcentagem.

Passando desses métodos de avaliação é possível partir para a geração da matriz de confusão, ela que por sua vez é uma ferramenta essencial para a avaliação de desempenho do modelo de classificação feito, conforme mostrado na **Figura** 32.

Figura 32 - Matriz de Confusão do Sistema 1.



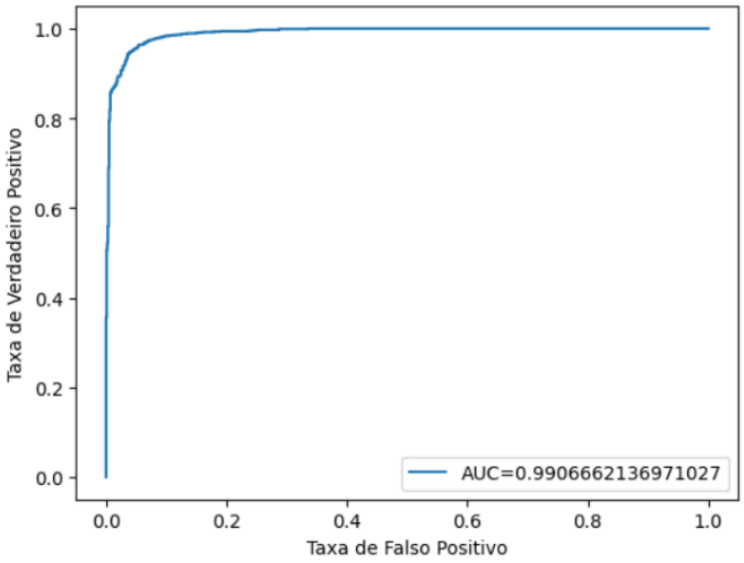
Fonte: O autor (2024).

Explicando a matriz de confusão da **Figura** 32, VP e VN representam as classificações corretas, enquanto que FP e FN representam os erros de classificação:

- Verdadeiros Negativos (VN): O valor na célula (0,0) é 904. Isso indica que o modelo acertou na previsão de 979 amostras como negativas (classe 0).
- Falsos Positivos (FP): O valor na célula (0,1) significa que o modelo errou na previsão de 99 amostras como positivas quando, na verdade, eram negativas.
- Falsos Negativos (FN): O valor na célula (1,0) indica que o modelo previu de forma errada 215 amostras como negativas quando, na verdade, eram positivas.

- Verdadeiros Positivos (VP): O valor na célula (1,1) significa que o modelo acertou a previsão de 789 amostras como positivas (classe 1).

Figura 33 – Curva ROC do Sistema 1



Fonte: O autor (2024).

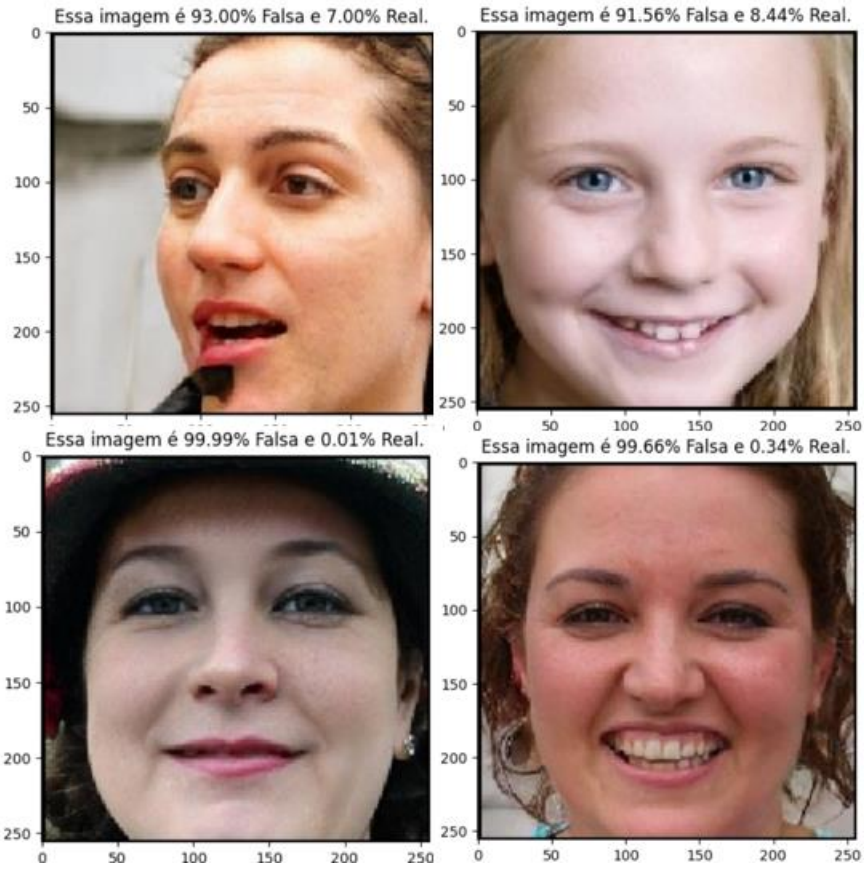
Uma das formas de avaliar os resultados do sistema é através da Curva ROC (*Receiver Operating Characteristic*), sendo bastante usada em na análise e avaliação de modelos de classificação em *Machine Learning* e estatísticas. Ela permite a visualização do desempenho do modelo ao mostrar a taxa de verdadeiros positivos, que são referentes a sensibilidade, no eixo Y e a taxa de falsos positivos, que são referentes e especifidade, no eixo X como mostrado na **Figura 33** acima.

O fato da curva se aproximar bastante do canto superior esquerdo, que é onde a sensibilidade é elevada e a taxa de falsos positivos é baixa, isso demonstra que o modelo tem a capacidade de distinguir bem entre as classes Verdadeiras e Falsas. Outra métrica importante que também é mostrada no gráfico da **Figura 33** é AUC (*Area Under the ROC Curve)*, o seu valor é de aproximadamente 0.99, o valor máximo que um modelo pode obtido é 1 sendo isso um indicativo de um modelo perfeito.

5.1.2 Resultado Qualitativos

Para avaliar a qualidade de classificação, foi feita uma função que recebe uma imagem com o parâmetro e retorna os resultados em porcentagens, os testes foram feitos com imagens do dataset separados para serem usados como teste.

Figura 34 – Resultados de classificação de imagens falsas



Fonte: O autor (2024).

As imagens da **Figura** 34 mostram os resultados da classificação da função, todas elas foram tiradas do diretório de imagens falsas, felizmente mostra como o modelo foi capaz de forma eficiente com mais de 90% de certeza em classificar que todas eram imagens falsas. Mas o contrário também é de extrema importância, como será mostrado na **Figura** 35.

É possível ver que o modelo não teve tanta certeza ao classificar o rosto da mulher na segunda coluna e linha da **Figura** 35, é possível ver que o rosto não foi adequadamente segmentado pelo Haarcascade, logo a imagem tem alguns elementos a mais coisa que pode afetar a classificação em um modelo treinado com poucas imagens, para poder classificar com mais eficiência pode ser necessário um banco de dados maior para o treinamento, além de acessórios, elementos como pinturas na face, marcas e cicatrizes podem afetar negativamente a classificação de faces.

Figura 35 - Resultados de classificação de imagens reais

Fonte: O autor (2024).

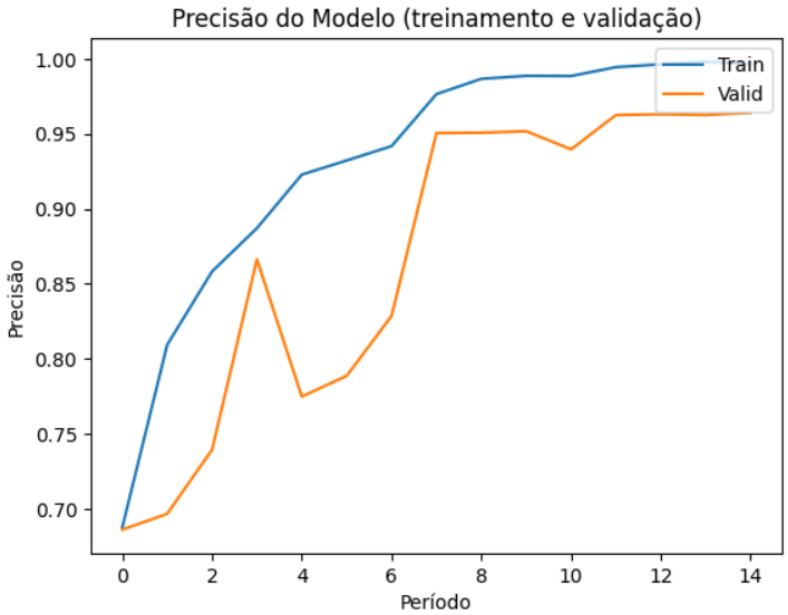
5.2 Resultados do Sistema 2

Aqui serão apresentados os resultados do Sistema 2, nele foram usadas 7953 imagens para o treino do modelo, 3978 imagens para a validação e 3983 para os testes, o *bach_size* foi igual a 32, portanto, o modelo precisou de 249 *backpropagations* em 15 épocas. Essa diferença no método de geração do modelo teve como objetivo uma melhora na sua capacidade de classificação.

5.2.1 Resultados Quantitativos

O gráfico da **Figura** 36 permite visualizar o desempenho do modelo do Sistema 2 durante o treinamento, é possível ver uma melhora dos valores de treinamento, a precisão de treino do sistema oscila entre 0.7 e 1.0, o valor inicial é significativamente maior que os do treinamento do Sistema 1.

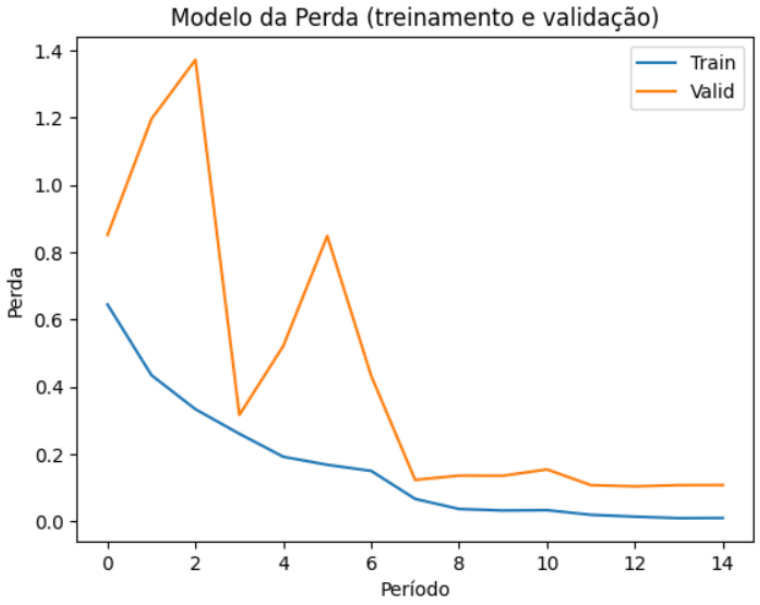
Figura 36 - Gráfico de Precisão do Modelo do Sistema 2



Fonte: O autor (2024).

Observando o gráfico é perceptível que após o 3° período, há uma queda significativa na precisão da validação que somente volta a crescer depois do 4° período, enquanto a precisão do treinamento continua a aumentar. Isso sugere novamente um fenômeno de *overfitting*, o modelo se ajustou muito bem aos dados de treinamento, mas não generalizou tão bem para os dados de validação. Para combater o overfitting, técnicas como regularização, parada antecipada (*early stopping*) e aumento de dados (*data augmentation*) podem ser empregadas.

Figura 37 - Gráfico de Perda do Modelo do Sistema 2



Fonte: O autor (2024).

A **Figura** 37 mostra o problema do *overfitting* sob outra perspectiva. Usando a função "*evaluate()*" do *Tensorflow* os resultados dessa avaliação são mostrados na **Figura** 38.

Figura 38 - Verificação com os dados de teste

```
# Avalia o modelo
test_loss, test_acc = model.evaluate(test_ds)
print("Teste Perda:", test_loss)
print("Test Precisão:", test_acc)
```

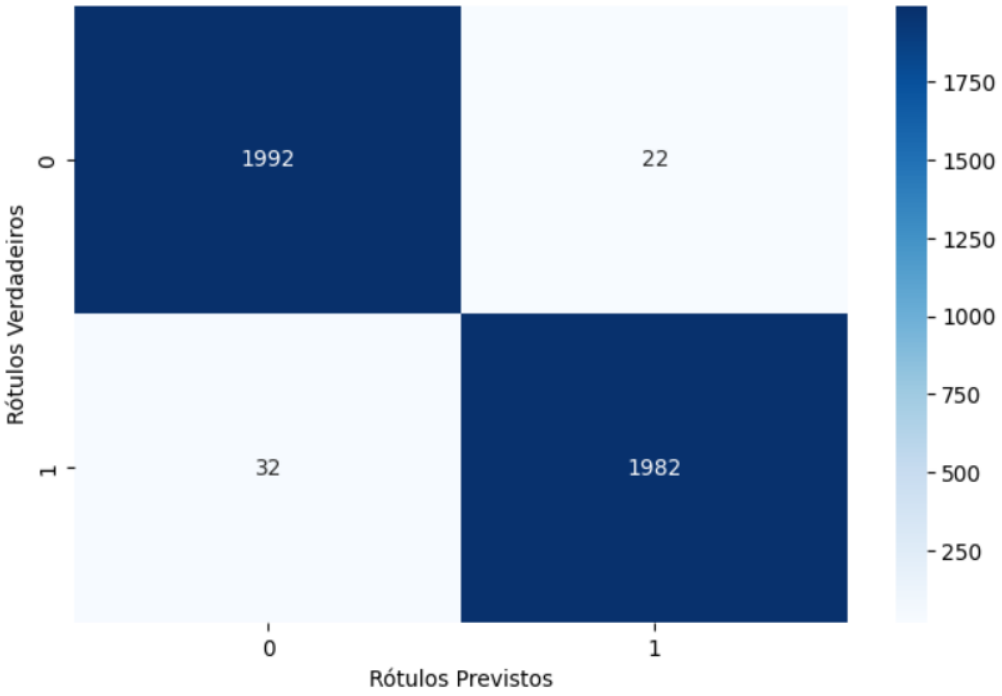
```
63/63 [==============================] - 18s 234ms/step - loss: 0.0760 - accuracy: 0.9696
Teste Perda: 0.07597245275974274
Test Precisão: 0.9696208834648132
```

Fonte: O autor (2024).

Como mostrado na **Figura** 38, mesmo com o problema do *overfitting* o sistema teve um nível de precisão de 96.96% para os dados de teste. Para casos de necessidade em melhorar ainda mais o modelo, a simples tarefa de carregar os pesos novamente pode melhorar essa porcentagem.

Passando desses métodos de avaliação, podemos partir para a geração da matriz de confusão para avaliar o desempenho do modelo de classificação do Sistema 2 conforme mostrado na **Figura** 39.

Figura 39 - Matriz de Confusão do Sistema 2



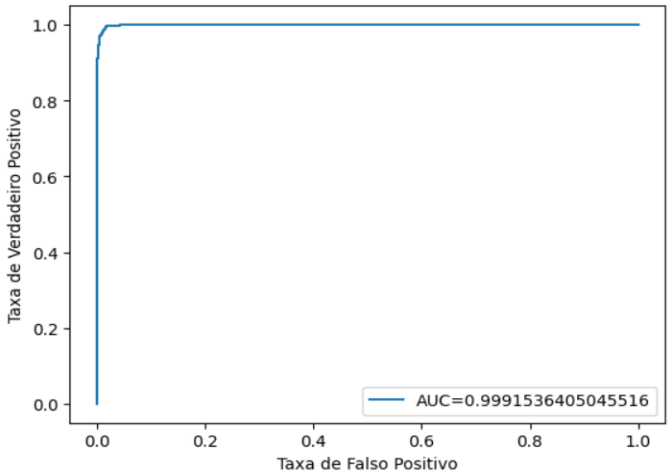
Fonte: O autor (2024).

Como mostrado nos gráficos anteriores é perceptível a melhora em usar um conjunto de imagens maior para o treinamento do modelo. Em resumo os resultados mostrados pela matriz de confusão são:

- Verdadeiros Negativos (VN): acertou na previsão de 1992 amostras como negativas (classe 0).
- Falsos Positivos (FP): errou na previsão de 22 amostras como positivas quando, na verdade, eram negativas.

- Falsos Negativos (FN): errou na previsão de 32 amostras como negativas quando, na verdade, eram positivas.
- Verdadeiros Positivos (VP): acertou na previsão de 1982 amostras como positivas (classe 1).

Figura 40 – Curva ROC do Sistema 2



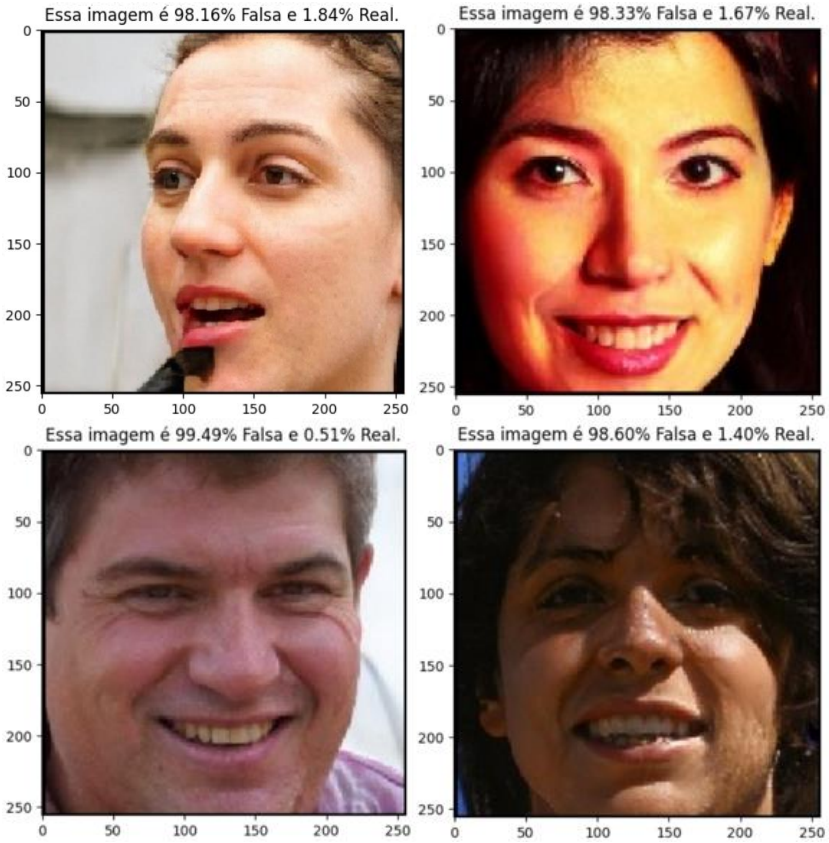
Fonte: O autor (2024).

E novamente para avaliar os resultados do Sistema 2 está sendo mostrado um gráfico na **Figura 40** da curva de ROC, ligeiramente diferente do Sistema 1 a curva do gráfico se aproxima do canto superior esquerdo de forma mais brusca, isso ilustra que o modelo tem a capacidade de distinguir com mais eficiência as classes Verdadeiras das Falsas. O valor do AUC também apresenta uma pequena mudança no valor de 0.99 sendo um pouco mais próximo do valor máximo de 1 que determina o quão próximo do modelo ideal é o modelo avaliado, possível para área abaixo da curva ROC.

5.2.2 Resultado Qualitativos

A seguir estão alguns resultados obtidos durante os testes de classificação, nas 4 primeiras imagens mostradas da **Figura** 41 são as imagens do grupo de falsas, enquanto que na **Figura 42** estão as imagens do grupo das verdadeiras.

Figura 41 - Classificações de imagens falsas feitas pelo sistema 2



Fonte: O autor (2024).

Todas as imagens da **Figura** 41 acima foram tiradas da pasta do diretório de testes de imagens fake, pode-se observar que todas ficaram com mais de 95% de certeza de serem falsas.

Esse mesmo fato acontece para as imagens verdadeiras mostradas na Figura 42, o sistema conseguiu classificar com mais de 99% de exatidão que as imagens eram verdadeiras. Até mesmo a imagem que não foi classificada de forma adequada no Sistema 1 teve uma melhor resposta, isso ilustra a diferença que o tamanho de dados faz no treinamento de um modelo.

Figura 42 - Classificação de imagens verdadeiras feitas pelo sistema 2

Fonte: O autor (2024).

Com os testes apresentados é possível considerar que os sistemas obtiveram resultados satisfatórios, em algumas imagens usadas para teste o algoritmo foi incapaz de reconhecer com precisão, variações de resultado assim quase sempre estão associadas a elementos a mais que não muito comuns nas imagens usadas no treino, tal com pinturas, acessórios e ângulos diferentes.

6 Considerações Finais

Esse trabalho se propôs a desenvolver um sistema classificador de faces geradas por inteligência artificial, o objetivo era distinguir rostos genuinamente humanos de rostos gerados por IA. Foi utilizada uma metodologia que incluiu levantamento bibliográfico, coleta e verificação de dados, treinamento do sistema, teste do software e comparação com outras literaturas.

A motivação para este estudo estava na necessidade de combater a disseminação de deepfakes e rostos gerados por IA, frequentemente utilizados em crimes virtuais. Portanto, a construção de uma ferramenta que pode contribuir para a prevenção dessas atividades ilícitas, sendo capaz de identificar e distinguir entre faces reais e sintéticas com eficácia seria de grande ajuda.

Com essas questões levantadas, para o desenvolvimento do projeto foi utilizado o Google Colab. pela questão de facilidade da configuração do ambiente, o projeto teve 140 mil fotos a disposição no dataset do Kaggle, mesmo assim não foi necessário o uso de todas elas, depois de feita a configuração do ambiente de programação foi necessário rotular todas as fotos separadas para o treinamento dos modelos, após esse processo de organização foi realizado o pré-processamento das imagens para que pudessem ser usadas no treinamento do modelo.

Depois que o modelo foi treinado, era necessário fazer as medições sobre o seu desempenho, foi detectado um certo grau de *overfitting*, que é quando o modelo se adequa bem aos dados de treinamento, mas tem dificuldade em assimilar os dados de validação, mesmo assim os dois sistemas tiveram um desempenho satisfatório sendo capazes de classificar corretamente a grande maioria das imagens de teste usadas.

6.1 Projetos Futuros

A Inteligência Artificial é um campo que está em constante evolução em com uma infinita capacidade de atuação, todas as áreas humanas podem ser complementadas com a IA em algum nível, a área de geração de imagens também está acompanhando esse constante crescimento e ficando mais evoluída ao ponto de ser capaz de gerar imagens cada vez mais realistas e quase indistinguíveis para a maioria das pessoas.

No futuro pretende-se o desenvolvimento de um Sistema Detector de Deepfakes em tempo real, torná-lo robusto e eficiente tendo a capacidade de identificar e distinguir entre faces autênticas de deepfakes com precisão, proporcionando uma defesa eficiente contra a disseminação de informações falsas e protegendo a integridade das comunicações digitais ao detectar uma tentativa de falsidade ideológica, o desenvolvimento de tal sistema pode vir a ser uma oportunidade comercial.

Referências

Qual é a importância de treinar um modelo para mais épocas no TensorFlow.js? Disponível em: <https://pt.eitca.org/inteligência-artificial/fundamentos-de-tensorflow-eitc-ai-tff/tensorflow-js/tensorflow-js-no-seu-navegador/revisão-de-exame-tensorflow-js-no-seu-navegador/qual-é-o-significado-de-treinar-um-modelo-para-mais-épocas-no-tensorflow-js/>. Acesso em: 17 fev 2024.

AGARWAL, Shivang e TERRAIL, Jean Ogier Du e JURIE, Frédéric. **Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks**. 2019. Normandie Univ, UNICAEN, ENSICAEN, CNRS, 2019. Disponível em: <http://arxiv.org/abs/1809.03193>.

ALPAYDIN, Ethem. **Introduction to machine learning**. [S.l.]: MIT press, 2020.

ARADI, Szilárd. **Survey of deep reinforcement learning for motion planning of autonomous vehicles**. IEEE Transactions on Intelligent Transportation Systems, v. 23, n. 2, p. 740–759, 2020.

BAKSHI, Urvashi e SINGHAL, Rohit. **A SURVEY ON FACE DETECTION METHODS AND FEATURE EXTRACTION TECHNIQUES OF FACE RECOGNITION**. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), v. 3, n. 3, Maio 2014. Disponível em: <www.ijettcs.org>.

BRÊTAS, Pollyanna. **Golpe: 1 em cada 4 brasileiros que tiveram relacionamentos virtuais diz já ter sido vítima de perfil falso, mostra pesquisa**. Disponível em: <https://extra.globo.com/economia-e-financas/golpe-1-em-cada-4-brasileiros-que-tiveram-relacionamentos-virtuais-diz-ja-ter-sido-vitima-de-perfil-falso-mostra-pesquisa-25425827.html>. Acesso em: 4 jan 2024.

CALANCA, Paulo e MATHEUS, Yuri e RAPHAELL, Bruno. **Quais são os 4 tipos de aprendizagem na IA, algoritmos e usos no dia a dia**. Disponível em:

<https://www.alura.com.br/artigos/quais-sao-tipos-aprendizagem-ia-inteligencia-artificial>. Acesso em: 27 jan 2024.

CHANDRAKALA, Pothuraju e SRINIVAS, B. e KUMAR, M. Anil. **Real Time Face Detection and Face Recognition using OpenCV and Python**. Journal of Engineering Sciences, v. 13, 2022. Disponível em: <www.jespublication.com>.

D'ADDARIO, M e DE FARIA ZANANI, R T F. **Inteligência Artificial: Tratados, aplicações, usos e futuro**. [S.l.]: Babelcube Incorporated, 2022. Disponível em: <https://books.google.com.br/books?id=lZ6VEAAAQBAJ>.

DIWAN, Tausif e ANIRUDH, G. e TEMBHURNE, Jitendra V. **Object detection using YOLO: challenges, architectural successors, datasets and applications**. Multimedia Tools and Applications, v. 82, n. 6, p. 9243–9275, 1 Mar 2023.

FALCÃO, João Vitor Regis e colab. **Redes neurais deep learning com tensorflow**. RE3C-Revista Eletrônica Científica de Ciência da Computação, v. 14, n. 1, 2019.

GAVALI, Pralhad, BANU, J. Saira. **Deep Convolutional Neural Network for Image Classification on CUDA Platform**. Academic Press, p. 99–122, 2019.

GUNAWAN, Teddy Surya e colab. **Development of video-based emotion recognition using deep learning with Google Colab**. TELKOMNIKA (Telecommunication Computing Electronics and Control), v. 18, n. 5, p. 2463, 1 Out 2020. Disponível em: <http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/16717>. Acesso em: 18 fev 2024.

GUO, H e WANG, X e LYU, S. Detection of Real-Time Deepfakes in Video Conferencing with Active Probing and Corneal Reflection. 2023, [S.l: s.n.], 2023. p. 1–5. Disponível em: <https://arxiv.org/pdf/2210.14153.pdf>. Acesso em: 3 fev 2024.

GUO, Hui e colab. Eyes Tell All: Irregular Pupil Shapes Reveal GAN-Generated Faces. 23 May 2022, Singapore: IEEE, 23 May 2022. p. 2904–2908. Disponível em: <https://arxiv.org/pdf/2109.00162.pdf>. Acesso em: 5 fev 2024.

HU, Shu e LI, Yuezun e LYU, Siwei. Exposing GAN-generated faces using inconsistent corneal specular highlights. 2021, [S.l.]: ICASSP. IEEE, 2021. Disponível em: <https://arxiv.org/pdf/2009.11924.pdf>. Acesso em: 25 jan 2024.

HUANG, Gao e colab. **Densely Connected Convolutional Networks**. 24 Ago 2016. Disponível em: <https://arxiv.org/pdf/1608.06993.pdf>. Acesso em: 7 jan 2024.

IZBICKI, Rafael e DOS SANTOS, Tiago Mendonça. **Aprendizado de máquina: uma abordagem estatística**. [S.l.]: Rafael Izbicki, 2020.

JAGTAP, A. M. e KANGALE, Vrushabh e UNUNE, Kushal. **A Study of LBPH, Eigenface, Fisherface and Haar- like features for Face recognition using OpenCV**. International Conference on Intelligent Sustainable Systems (ICISS 2019). [S.l.]: Institute of Electrical and Electronics Engineers. 2019

KAMENCAY, Patrik e colab. **A new method for face recognition using convolutional neural network**. Advances in Electrical and Electronic Engineering, v. 15, n. 4 Special Issue, p. 663–672, 2017.

KELANA, Muhammad Hilmi Bin. **FACIAL RECOGNITION AT GATED COMMUNITY**. 2021. Dissertation – Universiti Teknologi PETRONAS, 2021.

KELLY M.D. **Visual Identification of People by Computer**. 1970. Stanford AI Project, Stanford, CA, USA., 1970.

KORTLI, Yassin e colab. **Face recognition systems: A survey**. Sensors (Switzerland). [S.l.]: MDPI AG. Disponível em: <https://www.mdpi.com/1424-8220/20/2/342>. Acesso em: 6 jan 2024. 2 Jan 2020

LI, Lixiang e colab. **A Review of Face Recognition Technology**. IEEE Access, v. 8, p. 139110–139120, 2020.

LUDERMIR, Teresa Bernarda. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências**. Estudos Avançados, v. 35, p. 85–94, 2021. Disponível em: <https://www.scielo.br/j/ea/a/wXBdv8yHBV9xHz8qG5RCgZd/?lang=pt&format=pdf>. Acesso em: 1 fev 2024.

MOORE, M. **Fake accounts on social media, epistemic uncertainty and the need for an independent auditing of accounts.**

NAKAGAWA, Liliane. **Python é considerada a linguagem preferida pelo segundo ano consecutivo, segundo Tiobe**. Disponível em: <https://tecmasters.com.br/python-considerada-preferida-segundo-ano/>. Acesso em: 3 jan 2024.

PORKODI, S. P. e colab. **Generic image application using GANs (Generative Adversarial Networks): A Review**. Evolving Systems. [S.l.]: Institute for Ionics. , 1 Out 2023

ROSEBROCK, Adrian. **OpenCV Haar Cascades**. Disponível em: <https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/>. Acesso em: 2 mar 2024.

RUDEK, M.; COELHO, L. S.; CANCIGLIERI J. O. **Visão Computacional Aplicada a Sistemas Produtivos: Fundamentos e Estudo de Caso**. 2001. Pontifícia Universidade Católica do Paraná, PUCPR/LAS/CCET, Curitiba, 2001.

SIGUT, Jose e colab. **OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts**. IEEE Transactions on Education, v. 63, n. 4, p. 328–335, 1 Nov 2020.

SMACH, F.; MITERAN, J.; ATRI, M.; DUBOIS, J.; ABID, M.; GAUTHIER, J.P. **An FPGA-based accelerator for Fourier Descriptors computing for color object recognition using SVM.** . p. 249–258, 2007.

STONE, Zak e ZICKLER, Todd e DARRELL, Trevor. Autotagging facebook: Social network context improves photo annotation. 2008, [S.l.]: IEEE, 2008. p. 1–8.

TAGIAROLI, Guilherme. **Ele achou mais de 100 perfis fakes com sua foto: "cansei"; veja o que fazer..**. Disponível em: <https://www.uol.com.br/tilt/noticias/redacao/2023/02/03/perfis-fakes-redes-sociais-o-que-fzer.htm#:~:text=Segundo%20Solando%2C%20da%20OAB-SP,de%20ocorrência%2C%20até%20como%20precaução.>. Acesso em: 4 jan 2024.

TENSORFLOW. **Noções Básicas de ML com o a Keras**. Disponível em: <https://www.tensorflow.org/guide/keras?hl=pt-br>. Acesso em: 18 fev 2024.

THAI, Le Hoang e HAI, Tran Son e THUY, Nguyen Thanh. **Image Classification using Support Vector Machine and Artificial Neural Network**. International Journal of Information Technology and Computer Science, v. 4, n. 5, p. 32–38, 2 Maio 2012.

VIOLA, Paul e JONES, Michael. Rapid object detection using a boosted cascade of simple features. 2001, [S.l.]: Ieee, 2001. p. I–I. Disponível em: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>. Acesso em: 2 mar 2024.

YANG, Shuo e colab. **WIDER FACE: A Face Detection Benchmark**. . Hong Kong: [s.n.], 2016. Disponível em: <http://mmlab.ie.cuhk.edu.hk/projects/>.

ZARE, Reza e POURKAZEMI, Arash. **DenseNet approach to segmentation and classification of dermatoscopic skin lesions images**. arXiv preprint arXiv:2110.04632, 2021.

ZHANG, Jonathan e colab. **Classification of diabetic retinopathy severity in fundus images with DenseNet121 and ResNet50**. arXiv preprint arXiv:2108.08473, 2021.

ZHOU, L.; PAN, S.; WANG, J.; VASILAKOS, A.V. **Machine learning on big data: Opportunities and challenges.** Neurocomputing 2017, p. 350–361, 2017.

ZHU, Xiaojin Jerry. **Semi-supervised learning literature survey**. 2005. Disponível em: <https://minds.wisconsin.edu/bitstream/handle/1793/60444/TR1530.pdf?sequence=1&isAllowed=y>. Acesso em: 27 jan 2024.