UNIP - UNIVERSIDADE PAULISTA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

AUGUSTO CESAR MARCONDES DE OLIVEIRA

BRUNO SILVA RAMOS

GABRIEL DE OLIVEIRA

GABRIEL SIQUEIRA MARINHO

ISABELLA MARIA GASPAR ESPINDOLA FÉLIX

LUCAS CESAR FERREIRA

RAFAELLE CRISTINE SANTOS DA SILVA

ANÁLISE E DESENVOLVIMENTO DE UM SISTEMA PARA IDENTIFICAÇÃO DE VÍDEOS DEEPFAKE

SÃO JOSÉ DOS CAMPOS
2025

AUGUSTO CESAR MARCONDES DE OLIVEIRA – RA: N03204-9

BRUNO SILVA RAMOS: N84913-4

GABRIEL DE OLIVEIRA – RA: G460GA-0

GABRIEL SIQUEIRA MARINHO – RA: N03748-2

ISABELLA MARIA GASPAR ESPINDOLA FÉLIX - RA: N80846-2

LUCAS CESAR FERREIRA – RA: T65385-1

RAFAELLE CRISTINE SANTOS DA SILVA – RA: F349DH-4

ANÁLISE E DESENVOLVIMENTO DE UM SISTEMA PARA IDENTIFICAÇÃO DE VÍDEOS DEEPFAKE

Trabalho de Curso apresentado ao Instituto de Ciências Exatas e Tecnologia da Universidade Paulista, como parte dos requisitos necessários para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. MSc. André Yoshimi Kusumoto.
Coorientador: Prof. Dr. Fernando Mauro de Souza.

SÃO JOSÉ DOS CAMPOS

2025

CIP - Catalogação na Publicação

ANÁLISE E DESENVOLVIEMNTO DE UM SISTEMA PARA IDENTIFICAÇÃO DE VÍDEOS DEEPFAKE / Gabriel Oliveira...[et al.]. - 2025.
79 f. : il.

Trabalho de Conclusão de Curso (Graduação) apresentado ao Instituto de Ciência Exatas e Tecnologia da Universidade Paulista, São José dos Campos, 2025.
Área de Concentração: Aprendizado de Máquina Aplicado à Análise de Vídeos.
Orientador: Prof. Me. André Kusumoto.
Coorientador: Prof. Dr. Fernando Souza.

1. Detecção de Deepfakes. 2. Aprendizado de Máquina. 3. Redes Neurais Convolucionais. 4. Visão Computacional. 5. Processamento de Imagens e Vídeos. I. Oliveira, Gabriel. II. Kusumoto, André (orientador). III. Souza, Fernando (coorientador).

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Universidade Paulista com os dados fornecidos pelo(a) autor(a).

AUGUSTO CESAR MARCONDES DE OLIVEIRA

BRUNO SILVA RAMOS

GABRIEL DE OLIVEIRA

GABRIEL SIQUEIRA MARINHO

ISABELLA MARIA GASPAR ESPINDOLA FÉLIX

LUCAS CESAR FERREIRA

RAFAELLE CRISTINE SANTOS DA SILVA

ANÁLISE E DESENVOLVIMENTO DE UM SISTEMA PARA IDENTIFICAÇÃO DE VÍDEOS DEEPFAKE

Trabalho de Curso apresentado ao Instituto de Ciências Exatas e Tecnologia da Universidade Paulista, como parte dos requisitos necessários para a obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. MSc. André Yoshimi Kusumoto

O trabalho foi considerado Aprovado com a nota nove (9,0).
São José dos Campos, 12 de Novembro 2025.

André Kusumoto 13/11/2025

Prof. MSc. André Yoshimi Kusumoto – Orientador
Universidade Paulista – UNIP

Thiago José dos SF 13/11/25

Prof. MSc. Thiago José dos Santos Freitas – Avaliador
Universidade Paulista - UNIP

AGRADECIMENTOS

Agradecemos primeiramente a Deus, pela força, sabedoria e determinação que nos acompanharam durante toda esta jornada. Sua presença nos deu serenidade para superar os desafios e alcançar nossos objetivos.

Aos nossos familiares, expressamos profunda gratidão pelo apoio incondicional, paciência e incentivo contínuo. Aos amigos, agradecemos pela companhia, compreensão e por tornarem esta jornada mais leve e motivadora.

Estendemos um especial agradecimento ao nosso orientador, Prof. MSc. André Yoshimi Kusumoto, por sua dedicação, paciência e pelas orientações valiosas que contribuíram de forma decisiva para o desenvolvimento deste trabalho.

Também agradecemos ao nosso coorientador, Prof. Dr. Fernando Mauro de Souza, pelas contribuições técnicas, pelas sugestões assertivas e por todo o suporte durante o processo de pesquisa e execução do projeto.

Agradecemos ainda a todos os professores do curso de Ciência da Computação da Universidade UNIP, que contribuíram de maneira significativa para nossa formação, transmitindo não apenas conhecimento técnico, mas também valores de ética, responsabilidade e comprometimento profissional. Cada disciplina e orientação recebida foram essenciais para a construção do aprendizado que culmina neste projeto.

Por fim, agradecemos a todos que, direta ou indiretamente, participaram desta trajetória, seja com palavras de incentivo, ideias, críticas construtivas ou apoio direto, tornando possível a realização deste trabalho e o fechamento de mais uma importante etapa de nossas vidas.

RESUMO

O trabalho aborda o crescimento das tecnologias de geração de imagens sintéticas, com foco nas *deepfakes*, que representam uma ameaça crescente à autenticidade de conteúdos digitais e à segurança. O problema central está na facilidade de disseminação de vídeos manipulados, o que dificulta a identificação por usuários comuns e até mesmo por sistemas tradicionais de verificação.

Como objetivo principal, o projeto visa desenvolver e avaliar uma ferramenta capaz de identificar manipulações faciais em vídeos, oferecendo uma análise baseada em score de confiabilidade para facilitar a interpretação do usuário.

A metodologia adotada consistiu na adaptação e implantação de uma inteligência artificial previamente treinada e vencedora de uma competição na plataforma *Kaggle*. Essa IA foi portada para um aplicativo que desenvolvemos, onde permite que o usuário carregue uma imagem para ser analisada. Foram realizados testes com vídeos reais e falsos obtidos na internet, variando em resolução, compressão e condições de captura. A análise considerou precisão, estabilidade dos resultados e impacto do tempo de processamento em diferentes dispositivos.

Como considerações finais, os experimentos demonstraram alta taxa de acerto, em vídeos nos quais o *deepfake* estava restrito apenas ao rosto, conseguindo identificar com precisão as manipulações. No entanto, quando aplicada a vídeos mais recentes, em que a IA interagia com cenários complexos ou múltiplos elementos, seu desempenho caiu significativamente. Observou-se que fatores como compressão excessiva e baixa luminosidade também impactam a análise, mas não comprometem de forma significativa a eficácia do sistema.

Palavras-chave: *DeepFake*; Inteligência Artificial; Detecção; Segurança; Análise.

ABSTRACT

The work addresses the growth of synthetic image generation technologies, with a focus on deepfakes, which represent an increasing threat to the authenticity of digital content and security. The central problem lies in the ease of dissemination of manipulated videos, making identification difficult for regular users and even for traditional verification systems.

The main objective of the project is to develop and evaluate a tool capable of identifying facial manipulations in videos, providing analysis based on a reliability score to facilitate user interpretation.

The methodology adopted consisted of adapting and deploying pre-trained artificial intelligence, which had previously won a competition on the Kaggle platform. This AI was integrated into an application we developed, allowing users to upload a video for analysis. Tests were conducted with both real and fake videos obtained from the internet, varying in resolution, compression, and capture conditions. The analysis considered accuracy, stability of results, and the impact of processing time on different devices.

As final considerations, the experiments demonstrated a high success rate in videos where the deepfake was limited to the face, accurately identifying manipulations. However, when applied to more recent videos, in which the AI interacted with complex scenarios or multiple elements, its performance dropped significantly. Factors such as excessive compression and low lighting were also observed to impact the analysis, but they do not significantly compromise the system’s overall effectiveness.

Keywords: Deepfake; Artificial Intelligence; Detection; Security; Analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AUC	*Area Under the Curve*
AWS	*Amazon Web Services*
CNNs	*Convolutional Neural Networks*
CWI	*Centrum Wiskunde & Informatica*
DFDC	*Deepfake Delection Challenge*
GAN	Redes Adversárias Generativas
GPUs	*Graphic Processing Units*
IA	Inteligência Artificial

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos, a inteligência artificial (IA) tem se tornado cada vez mais presente em diversas áreas, como saúde, educação e finanças, além de estar integrada a assistentes virtuais e outras tecnologias do dia a dia. Sua capacidade de automatizar tarefas, identificar padrões e realizar previsões tem impulsionado avanços significativos, transformando a forma como lidamos com a tecnologia (NORION, s.d).

Com ela podemos desenvolver tecnologias capazes de executar funções que normalmente necessitariam de conhecimento humano. Utilizando algoritmos e análise de dados, essas máquinas são projetadas para aprender, raciocinar e ter autonomia em suas decisões, aprimorando continuamente seu desempenho (NORION, s.d).

Os impactos da inteligência artificial são cada vez mais evidentes. Muitas funções que antes eram desempenhadas por seres humanos estão sendo substituídas por máquinas inteligentes, especialmente em tarefas repetitivas ou de alto risco. Por outro lado, a IA tem revolucionado diversos setores, como a mobilidade, com carros autônomos que prometem viagens mais seguras e a educação, ao proporcionar novos métodos de ensino adaptativos (AGÊNCIA SP, 2025). Na área da saúde, a IA tem se destacado pela capacidade de identificar doenças de forma precoce e precisa, além de recomendar tratamentos personalizados, aumentando a eficiência dos sistemas de atendimento médico (VINICIUS, 2023).

Contudo, a incorporação da IA nas atividades cotidianas evidencia novos dilemas éticos, impulsionando discussões sobre a necessidade de regulamentação e parâmetros normativos bem estruturados, principalmente no que diz respeito ao uso e possível exploração de dados pessoais. A mesma tecnologia capaz de otimizar processos, salvar vidas e gerar soluções inovadoras, se aplicada de forma inadequada, pode comprometer valores fundamentais e abrir espaço para o uso indevido de dados e manipulação de informações. Diante disso, cresce a necessidade de estabelecer diretrizes claras e regulamentações que orientem o desenvolvimento e a aplicação dessas ferramentas, garantindo que o progresso tecnológico caminhe junto com a preservação de princípios sociais (LBCA, 2024). Precisamos assegurar que seus benefícios sejam usados de maneira justa e responsável.

A inteligência artificial não está imune ao uso indevido por indivíduos mal-intencionados. Os ciberataques são um exemplo evidente de como a tecnologia pode

ser explorada de forma prejudicial, resultando em danos significativos tanto para indivíduos quanto para organizações.

Com a evolução dessa tecnologia, também aumentam os desafios e preocupações éticas especialmente no campo de criação e manipulação de conteúdo digital.

Um dos avanços que a IA trouxe é no campo de manipulação de imagens, chamadas *deepfakes*, que são mídias ultra realísticas de imagens, áudios ou vídeos manipulados para criar uma falsa representação. Os *deepfakes* fazem o uso de algoritmos de *deep learning*, principalmente as Redes Adversárias Generativas (GAN, s.d), para produzir mídias sintéticas que podem ser indistinguíveis da realidade. Essas *deepfakes* comumente são usadas em imagens de celebridades, políticos ou pessoas influentes, mas podem afetar também pessoas comuns, levando a espalhar a desinformação (LBCA, 2024).

A rápida proliferação do uso de *deepfake* tem levantado várias preocupações, especialmente na autenticidade em mídias visuais digitais. Imagens e vídeos são ferramentas de comunicação e persuasão comumente usadas, e a possibilidade de alterar seu conteúdo com tamanha precisão tem levantado dúvidas sobre o que é real e o que é fabricado. Isso resulta em uma perda de confiança no conteúdo visual, que impacta em opinião pública.

1.1 Motivação

A inteligência artificial tem evoluído muito nos últimos anos, proporcionado inovações significativas, mas também trouxe desafios preocupantes, como a disseminação de *deepfakes*. Essas mídias sintéticas, geradas por redes neurais avançadas, conseguem manipular imagens, vídeos e áudios de maneira extremamente realista, dificultando a distinção entre conteúdo autêntico e falsificado. O impacto dessa tecnologia vai além do entretenimento, tornando-se uma ameaça em diversas áreas, incluindo política, segurança cibernética e desinformação digital.

Nos últimos anos, diversos casos de *deepfakes* ganharam destaque e levantaram preocupações sobre os riscos dessa tecnologia. Um exemplo marcante foi um vídeo manipulado do ex-presidente dos Estados Unidos, Barack Obama, no qual ele aparentemente fazia um discurso polêmico e ofensivo, como mostra a reportagem (EDUCACÃO E MÍDIA, 2020). Outro caso notório envolve a criação de vídeos falsos

de celebridades em situações constrangedoras, contribuindo para a propagação de desinformação e difamação (E-SAFER, s.d).

Casos de uso criminoso de deepfakes têm ganhado destaque, incluindo situações de extorsão envolvendo grandes corporações. Um exemplo ocorreu na China, quando uma multinacional foi alvo de um golpe estimado em US$ 25,6 milhões (aproximadamente R$ 127 milhões). Nesse esquema, criminosos utilizaram deepfakes para simular a imagem e a voz do diretor financeiro da empresa durante uma videoconferência, solicitando transferências bancárias. Relatórios apontam que todos os participantes da reunião, com exceção da vítima, eram representações falsas de pessoas criadas digitalmente (FORBES, 2024).

Isso mostra que o *deepfake* representa tanto um avanço tecnológico quanto um risco significativo. Para lidar de forma que as informações sejam confiáveis, teremos que contar com o desenvolvimento tecnológico de análise forense computacional moderna para que eles possam detectar os vídeos e áudios manipulados (E-SAFER, s.d).

Diante desse cenário, a criação de um sistema capaz de detectar *deepfakes* se mostra essencial para combater fraudes digitais e preservar a integridade da informação. A falta de ferramentas eficazes para identificar esse tipo de manipulação coloca em risco a credibilidade de notícias, processos judiciais e até a privacidade das pessoas. Além disso, como vimos, empresas e instituições podem ser alvo de ataques que exploram essas falsificações para disseminar informações enganosas ou realizar fraudes financeiras.

Assim, este trabalho tem o objetivo de contribuir para o desenvolvimento de uma solução tecnológica voltada à identificação de vídeos *deepfakes,* auxiliando na mitigação dos riscos associados a essa prática. A justificativa do tema se dá pela necessidade urgente de mecanismos que garantam a confiabilidade da informação no ambiente digital, fortalecendo a segurança e promovendo o uso ético da inteligência artificial.

1.2 Definição do problema

O surgimento de vídeos *deepfake*, produzidos por técnicas avançadas de inteligência artificial, tornou possível criar conteúdo falsos altamente realistas e difíceis de identificar a olho nu. Embora possam ter aplicações legítimas, seu uso indevido

tem causado riscos à privacidade, à reputação e à disseminação de informações falsas.

A detecção manual desses vídeos é limitada e ineficiente diante do rápido avanço da tecnologia. Assim, este projeto busca desenvolver um método automatizado capaz de identificar *deepfakes* com precisão, contribuindo para a segurança da informação e a verificação de conteúdos audiovisuais.

1.3 Objeto Geral

Este trabalho tem como objetivo geral identificar o uso de *deepfakes* e técnicas de manipulação digital em vídeos contendo faces humanas geradas por inteligência artificial, por meio da aplicação de métodos baseados em *deep learning*. E busca-se atingir isso a partir do desenvolvimento de uma ferramenta, que através de um aplicativo será capaz de reconhecer alterações sintéticas em imagens faciais, promovendo maior segurança e confiabilidade na verificação da autenticidade visual em ambiente digitais.

1.4 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Investigar as principais técnicas de deepfakes e os mecanismos de manipulação facial baseados em inteligência artificial;
- Analisar arquiteturas de redes neurais utilizadas na detecção de mídias sintéticas, com foco em **modelos pré-treinados**;
- Adaptar e preparar um conjunto de dados contendo vídeos reais e manipulados para validação do modelo escolhido;
- Implementar e integrar o modelo de detecção de deepfake em um aplicativo funcional, permitindo testes práticos de análise em tempo real;
- Avaliar o desempenho do sistema desenvolvido por meio de métricas como tempo de processamento, score de autenticidade e taxa de acerto, considerando diferentes qualidades de vídeo e dispositivos de execução;

- Comparar a eficácia do modelo em cenários reais, utilizando deepfakes encontrados na internet e vídeos capturados por diferentes aparelhos.

2. REVISÃO BIBLIOGRAFICA

Esta seção traz todas as referências bibliográficas necessárias para o desenvolvimento do presente trabalho.

2.1 Conhecimentos Básicos

2.1.1 Python

Python é uma linguagem de programação de alto nível amplamente reconhecida por sua clareza sintática e versatilidade. Criada por Guido Van Rossum em 1991, a linguagem destaca-se pela facilidade de leitura e pela extensa comunidade de desenvolvedores que contribuem para sua constante evolução (VAN ROSSUM, 1991). Seu uso é abrangente, contemplando áreas como desenvolvimento web, automação, análise de dados e inteligência artificial.

O desenvolvimento do Python teve início no final da década de 1980, enquanto van Rossum atuava no *Centrum Wiskunde & Informatica* (CWI), nos Países Baixos. Inspirado pela linguagem ABC, seu objetivo era criar uma alternativa mais intuitiva e eficiente. O primeiro lançamento oficial ocorreu em 1991 e, ao longo dos anos, a linguagem passou por diversas melhorias, consolidando-se como uma das mais utilizadas globalmente.

Um dos fatores que impulsionaram a adoção do Python é sua abrangente biblioteca padrão, além da vasta quantidade de frameworks e bibliotecas externas que facilitam o desenvolvimento de soluções avançadas. No campo da inteligência artificial, ferramentas como *TensorFlow*, desenvolvido pelo Google, e *PyTorch*, da Meta, desempenham um papel essencial no treinamento e implementação de modelos de aprendizado profundo (LUTZ, 2013).

Outro aspecto relevante do Python é sua compatibilidade com múltiplos sistemas operacionais, permitindo a execução de códigos sem necessidade de

grandes adaptações. Essa característica tornou a linguagem a escolha preferida de cientistas de dados e pesquisadores que trabalham com aprendizado de máquina e inteligência artificial.

2.1.2 Kotlin

Desenvolvido pela *JetBrains* em 2011, Kotlin é uma linguagem de programação moderna e de tipagem estática. Foi feita para ser concisa, segura e compatível com Java, tornando-se rapidamente uma das escolhas favoritas para o desenvolvimento em aplicativos Android. Sua sintaxe clara e expressiva simplifica a escrita de código, aumentando a produtividade e reduzindo a ocorrência de erros (JETBRAINS, 2011).

O projeto Kotlin teve início em 2010, quando a JetBrains buscava criar uma opção mais eficiente ao Java, mantendo a compatibilidade com o ecossistema da Java Virtual Machine (JVM). O nome da linguagem é uma homenagem à Ilha de Kotlin, próxima a São Petersburgo, na Rússia. Após um período de desenvolvimento contínuo, o Google anunciou oficialmente o suporte ao Kotlin para Android em 2017, consolidando sua posição como uma das principais linguagens para desenvolvimento mobile (GOOGLE DEVELOPERS, 2017).

Entre os pontos fortes do Kotlin estão a inferência de tipos, a segurança contra valores nulos (*null safety*), as funções de extensão e a integração perfeita com bibliotecas Java. Esses recursos contribuem para a criação de aplicações mais seguras e legíveis, preservando a compatibilidade com projetos já existentes. Além disso, o Kotlin oferece suporte tanto à programação funcional quanto à programação orientada a objetos, ampliando sua versatilidade (NAFZIGER; JEMEROV, 2017).

Uma das principais vantagens é o Kotlin *Multiplatform*, que permite o compartilhamento de código entre diversos sistemas operacionais, como Android, iOS, Windows e Linux. Esse recurso reforça o compromisso da linguagem com a portabilidade e o desenvolvimento moderno, combinando desempenho, produtividade e flexibilidade em um ecossistema unificado.

2.1.3 Deep Learning

O aprendizado profundo, também denominado *Deep Learning*, é um ramo do aprendizado de máquina que emprega redes neurais artificiais compostas por múltiplas camadas para analisar e interpretar grandes volumes de dados. Essa abordagem tem sido amplamente aplicada em tarefas como reconhecimento de imagens, processamento de linguagem natural e análise preditiva (GOODFELLOW et al., 2016). Embora a concepção das redes neurais artificiais tenha surgido na década de 1940, apenas com os avanços computacionais das últimas décadas essa tecnologia tornou-se viável em larga escala.

A história do *Deep Learning* remonta ao desenvolvimento do *Perceptron*, criado por Frank Rosenblatt em 1958. Apesar de promissor, esse modelo inicial possuía limitações e foi criticado na década de 1960, resultando em um período de menor interesse pela área. Nos anos 1980, com a evolução das redes neurais multicamadas e o desenvolvimento do algoritmo de retropropagação, a pesquisa na área foi retomada. Entretanto, apenas a partir dos anos 2000, com o avanço do poder computacional, especialmente das unidades de processamento gráfico (GPUs), o *Deep Learning* consolidou-se como a principal abordagem para inteligência artificial moderna.

As redes neurais profundas são compostas por múltiplas camadas de neurônios artificiais, cada uma responsável por transformar os dados de entrada em representações mais abstratas. Quanto maior o número de camadas, maior a capacidade da rede de identificar padrões complexos.

Componentes principais:

- **Camada de entrada**: recebe os dados (por exemplo, pixels de uma imagem);
- **Camadas ocultas**: realizam transformações nos dados utilizando pesos ajustáveis;
- **Camada de saída**: fornece o resultado, como a classificação de um objeto.

2.1.4 Aplicações do Deep Learning

O aprendizado profundo é amplamente empregado em diversas áreas, incluindo:

- Reconhecimento facial (tecnologias como Face ID e sistemas de vigilância);
- Tradução automática (exemplo: Google Tradutor, DeepL);
- Medicina (diagnóstico de doenças a partir de imagens);
- Veículos autônomos (Tesla, Waymo).

A principal vantagem do *Deep Learning* em relação a outras técnicas de aprendizado de máquina está em sua capacidade de reconhecer padrões complexos em grandes volumes de dados. A estrutura das redes neurais possibilita a transformação das informações de entrada em representações abstratas, tornando essa abordagem essencial para o desenvolvimento de tecnologias como reconhecimento facial, tradução automática e sistemas autônomos (LECUN, 2015; BENGIO, 2015; HINTON, 2015). O avanço do aprendizado profundo está diretamente associado ao aumento da capacidade computacional e à disponibilidade crescente de grandes volumes de dados. As GPUs desempenham um papel essencial nesse processo, permitindo o processamento paralelo de informações e acelerando o treinamento dos modelos neurais.

2.1.5 Deep Fake

A tecnologia conhecida como *Deep Fake* é baseada em aprendizado profundo e possibilita a criação de imagens e vídeos sintéticos altamente realistas. Utilizando técnicas como Redes Adversárias Generativas (GANs), essa tecnologia permite a manipulação de rostos e vozes em tempo real (GOODFELLOW et al., 2014). O termo "*Deep Fake*" tornou-se amplamente conhecido devido ao seu uso na criação de conteúdos enganosos, como vídeos de figuras públicas proferindo discursos ou realizando ações que nunca ocorreram.

As GANs operam por meio da interação entre duas redes neurais: o gerador, responsável pela criação das imagens falsas, e o discriminador, cuja função é avaliar a autenticidade dessas imagens. Esse processo de aprendizado contínuo permite que

o gerador produza conteúdos cada vez mais convincentes, tornando a detecção de falsificações um desafio.

Apesar de aplicações legítimas no setor cinematográfico e na criação de avatares digitais, o uso de *Deep Fakes* levanta sérias preocupações éticas e legais. A disseminação de vídeos manipulados pode ser explorada para fins políticos, fraudes e difamações, sendo essencial o desenvolvimento de ferramentas para detectar e mitigar esses impactos (CHESNEY; CITRON, 2019).

2.1.6 Estratégias para combater Deep Fakes

- **Ferramentas de detecção**: desenvolvimento de algoritmos capazes de identificar manipulações;
- **Educação e conscientização**: orientação da população sobre a identificação de falsificações;
- **Regulamentação**: criação de leis para limitar o uso indevido dessa tecnologia.

2.1.7 Considerações finais

A compreensão dos conceitos de *Python*, aprendizado profundo e *Deep Fake* é fundamental para o desenvolvimento deste Trabalho de Conclusão de Curso.

Cada uma dessas tecnologias exerce um impacto significativo na evolução da inteligência artificial e na sociedade contemporânea. O *Python* estabelece a base para a criação de soluções em aprendizado de máquina, enquanto o *Deep Learning* possibilita a construção de modelos sofisticados de inteligência artificial. No entanto, a tecnologia *Deep Fake* apresenta desafios éticos consideráveis, demandando um debate aprofundado sobre o uso responsável da inteligência artificial na produção e disseminação de informações.

2.2 Conhecimento de Ferramentas.

2.2.1 Open CV

O OpenCV (*Open Source Computer Vision Library*) é uma biblioteca livre que se destaca por seu uso em projetos de visão computacional, aprendizado de máquina e processamento de imagens digitais. Essa tecnologia disponibiliza uma grande variedade de recursos e algoritmos que facilitam a análise automatizada de elementos visuais, permitindo a identificação de rostos, objetos, padrões e movimentos de forma rápida e precisa.

Devido à sua ampla aplicabilidade, o OpenCV tornou-se uma ferramenta fundamental em soluções que utilizam inteligência artificial, sendo empregada em sistemas que realizam desde o monitoramento visual em tempo real até a interpretação de imagens complexas. Quando combinada com outros frameworks e bibliotecas, a ferramenta oferece maior desempenho e qualidade nos resultados de processamento visual, otimizando o desenvolvimento de aplicações modernas (ASIMOV ACADEMY, 2024).

2.2.2 FFmpeg

O FFmpeg é um framework capaz de fazer codificação, decodificação, transcodificar, multiplicar, desmultiplicar, transmitir, filtrar e reproduzir arquivos de multimidia. Suportando desde formatos mais antigos como mais atuais. Sendo uma solução multiplataforma e completa para gravar. Converter e transmitir áudio e vídeo (FFMPEG).

2.2.3 NumPy

O *NumPy* é uma das bibliotecas mais importantes para computação científica em *Python*, sendo amplamente utilizada para otimizar o processamento e a análise de dados. Sua principal característica é o uso de *arrays* e matrizes multidimensionais, que possibilitam a realização de operações complexas de maneira rápida e eficiente.

A biblioteca também disponibiliza um conjunto robusto de funcionalidades, como:

- Execução de operações matemáticas e lógicas;
- Manipulação e transformação de estruturas e formatos de dados;
- Classificação, filtragem e seleção de informações;
- Leitura e gravação de arquivos;
- Aplicação de funções estatísticas;
- Geração de números aleatórios para testes e simulações.

O *NumPy* é amplamente aplicado em projetos de *Machine Learning* e Ciência de Dados, integrando-se de forma nativa com outras ferramentas do ecossistema *Python*, como *Pandas*, *Matplotlib* e *SciPy*, o que o torna uma base essencial para o desenvolvimento de soluções analíticas e computacionais (HASH TAG TREINAMENTOS, 2025).

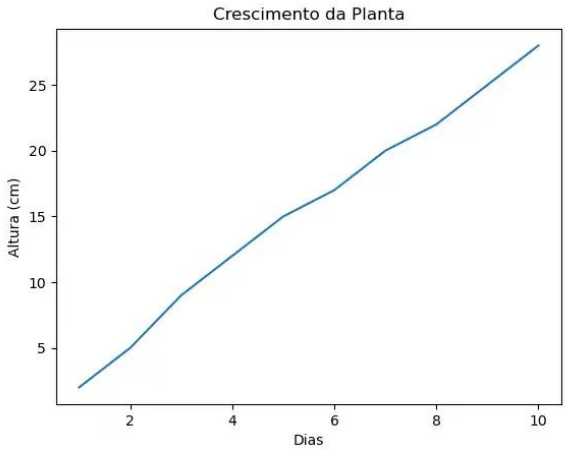
2.2.4 Matplotlib

O Matplotlib é uma biblioteca de visualização de dados em *Python*, amplamente utilizada para a criação de gráficos estáticos, animados e até mesmo em 3D. Reconhecida pela sua versatilidade, a ferramenta é empregada por cientistas de dados, engenheiros, pesquisadores e programadores em diferentes áreas.

A biblioteca disponibiliza uma interface simples e flexível, que permite a construção de representações gráficas variadas, como gráficos de linhas, gráficos de barras, gráficos de pizza, histogramas, dispersões, entre outros. Sua flexibilidade possibilita personalizações avançadas, desde a definição de estilos até a integração com outras bibliotecas como NumPy e Pandas, o que torna o processo de análise e apresentação de dados mais completo e eficiente (ANDRADE, 2023).

Na Figura 1, é possível observar um exemplo de visualização produzida com o Matplotlib.

Figura 1 - Exemplo de Interface com Gráfico de Linhas oferecido pelo Matplotlib



Autor: Desconhecido

2.2.5 FaceForensics++

O FaceForensics++ é um conjunto de dados desenvolvido para a detecção de manipulações faciais em vídeos. Ele consiste em 1.000 sequências de vídeos originais manipulados por quatro métodos diferentes automatizados: *DeepFakes, Face2Face, FaceSwap* e *NeuralTextures.* Os vídeos foram extraídos de 977 vídeos do YouTube, contendo rostos frontais rastreáveis, sem oclusões, permitindo que métodos automatizados de adulteração gerem forjas realistas. O conjunto de dados inclui máscaras binárias, o que possibilita a utilização para classificação de imagens e vídeos, bem como segmentação. Além disso, fornece 1.000 modelos *DeepFakes* para gerar e aumentar novos dados (RÖSSLER et al., 2019).

2.2.6 DFDC Dataset

O DFDC (*DeepFake Detection Challenge*) é um extenso conjunto de dados de rostos desenvolvido para treinar modelos de detecção de *deepfakes*, criado para a competição do Kaggle. Trata-se do maior *dataset* de manipulação facial atualmente disponível, contendo mais de 100.000 vídeos gerados a partir de 3.426 atores, utilizando diversos métodos de *deepfake*, incluindo técnicas baseadas em GANs e outros métodos não supervisionados. Apesar de a detecção de deepfakes ainda ser um desafio complexo e não totalmente resolvido, estudos mostram que modelos treinados exclusivamente com o DFDC podem generalizar (reconhecer padrões)

vídeos *deepfake* encontrados na internet, tornando-se uma ferramenta valiosa para a análise de vídeos potencialmente manipulados (DOLHANSKY; BITTON; PFLAUN; LU; HOWES; WANG; CANTON FERRER, 2020).

2.2.7 Celeb-DF

O Celeb-DF é um conjunto de dados de vídeos *deepfake* de alta qualidade, desenvolvido para aprimorar a detecção de manipulação de faces realistas. Composto por 5.639 vídeos de celebridades, totalizando mais de 2 milhões de quadros, o *dataset* foi projetado para refletir melhor as características dos vídeos *deepfake* encontrados na internet, superando limitações de *datasets* anteriores que apresentavam artefatos visuais evidentes.

Os vídeos reais foram coletados do Youtube, abrangendo mais de 59 celebridades de diferentes gêneros, idades e etnias. Sendo amplamente utilizado para treinar e avaliar algoritmos de detectação de *deepfakes*, contribuindo para o forense digital da segurança de dados (LI, 2020; VARGHESE, 2022).

2.2.8 Git e GitHub

O *Git* é um sistema de controle de versão e gerenciamento de projetos de software, desenvolvido por Linus Torvalds. Atualmente, praticamente todos os projetos de software utilizam o *Git* para gerenciar versões de código, possibilitando rastrear alterações, colaborar em equipe e manter o histórico completo do desenvolvimento (ATLASSIAN, 2025).

O GitHub é uma plataforma em nuvem em que é possível armazenar, compartilhar e trabalhar com outras pessoas para escrever códigos de softwares, armazenando os em um repositório (GITHUB, 2025).

2.2.9 MTCNN

O *Multitask Cascaded Convolutional Networks* (MTCNN) é um algoritmo amplamente utilizado para detecção e alinhamento facial, proposto originalmente em 2016 por pesquisadores como Kaipeng Zhang e colaboradores. Sua principal contribuição está na eficiência para localizar rostos em diferentes ângulos, tamanhos

e condições de iluminação, por meio de uma arquitetura composta por três redes neurais convolucionais dispostas em cascata.

Essas redes são organizadas em etapas:

- **P-Net (*Proposal Network*):** realiza uma varredura inicial da imagem em várias escalas, gerando regiões candidatas que podem conter rostos;
- **R-Net (*Refine Network*):** refina as regiões detectadas, eliminando falsos positivos e ajustando as caixas delimitadoras;
- **O-Net (*Output Network*):** faz a etapa final de refinamento e identifica cinco pontos de referência facial (olhos, nariz e canto da boca).

A MTCNN é amplamente reconhecida por combinar precisão e desempenho, sendo capaz de detectar rostos de forma confiável mesmo em situações desafiadoras, como variações de pose, expressão ou iluminação. Sua aplicação é comum em sistemas de segurança, reconhecimento facial, saúde, entretenimento e marketing, graças à sua robustez e velocidade na análise de imagens (MTCNN, s.d.; JAWABREH, 2023).

2.2.10 ML KIT

O ML Kit é um *Software Development Kit* (SDK) desenhado para integrar recursos de *machine learning* (aprendizado de máquina) dos dispositivos Google diretamente em aplicativos nativos para Android e iOS. Ele oferece uma coleção de interfaces de programação (APIs) poderosas e intuitivas nas áreas de Visão, Linguagem Natural e IA Generativa, permitindo que desenvolvedores resolvam problemas complexos ou criem experiências inovadoras (GOOGLE, 2025).

2.2.11 EfficientNet

EfficientNet é uma arquitetura de rede neural voltada para visão computacional que se destaca por equilibrar desempenho e eficiência. Diferente de modelos tradicionais, que aumentam profundidade ou resolução de forma isolada, essa arquitetura aplica um escalonamento uniforme entre largura, profundidade e tamanho da imagem, otimizando o uso dos recursos computacionais.

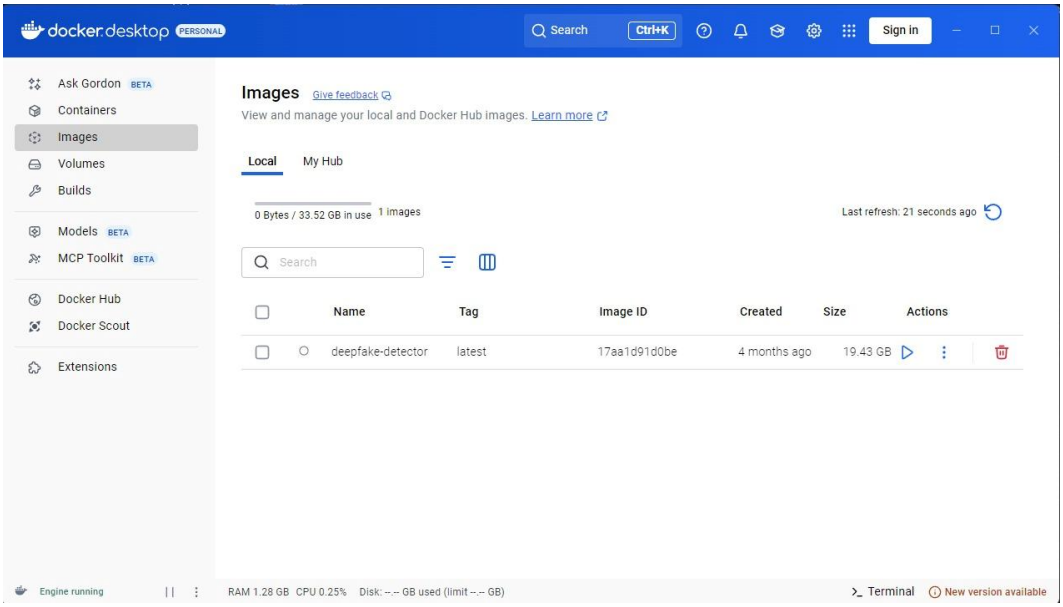
As Redes Neurais Convolucionais (CNNs) são amplamente utilizadas em tarefas como detecção facial e classificação de imagens, pois conseguem extrair padrões visuais diretamente das imagens de entrada. No entanto, conforme os conjuntos de dados se tornam mais complexos, essas redes precisam crescer em tamanho para manter a precisão, o que demanda mais processamento.

Esse aumento no custo computacional dificulta o uso de modelos tradicionais em dispositivos com hardware limitado, como smartphones. O *EfficientNet* surge justamente para contornar esse problema, oferecendo um modelo leve, escalável e adequado para aplicações em tempo real (KINGLER, 2024). Como o nosso projeto de detecção de *deepfake*

2.2.12 Docker

O Docker é uma plataforma de código aberto que possibilita aos desenvolvedores construir, implementar, executar, atualizar e gerenciar aplicações dentro de containers. Esses containers utilizam funcionalidades do kernel do Linux para isolar processos, permitindo que múltiplos aplicativos sejam executados de forma independente. Isso proporciona melhor aproveitamento da infraestrutura e mantém um alto nível de segurança, semelhante ao de sistemas isolados (SUSNJARA; SMALEY, s.d). Podemos verificar a plataforma na Figura 2.

Figura 2 - Interface Docker



Fonte: Autores (2025)

2.2.13 Android Studio

O *Android Studio* é uma IDE (ambiente de desenvolvimento integrado) oficial utilizado para desenvolvimento em apps Android. Ele usa como base o intelliJ DEA, como seu editor de código e suas ferramentas avançadas.

A ferramenta fornece um emulador integrado, facilitando a validação do aplicativo em dispositivos simulados, além de recursos de depuração e análise de desempenho, úteis para identificar gargalos e ajustar o uso de memória e processamento. Também oferece integração com controle de versão e suporte a bibliotecas externas, permitindo a incorporação de modelos de IA e ferramentas como *OpenCV*, *PyTorch Mobile* e outras (ANDROID, 2025).

3. METODOLOGIA

Nesta seção os métodos e procedimentos utilizados na criação deste trabalho serão descritos.

As etapas foram as seguintes:

- **Pesquisa Exploratória e Definição do Escopo**
    - Levantamento de notícias, artigos científicos e competições públicas (como a DFDC – *DeepFake Detection Challenge* do Kaggle) para compreender o impacto social e tecnológico das *deepfakes*.
    - Análise de *datasets* já disponíveis e ferramentas de detecção existentes para definir o diferencial da solução proposta.
    - Definição dos objetivos gerais e específicos do projeto com base na viabilidade técnica e relevância do tema.

- **Seleção do Modelo de IA**
    - Escolha de um modelo pré-treinado de detecção de *deepfakes*, vencedor da competição DFDC Kaggle.
    - Estudo da arquitetura do modelo para uso em ambiente de aplicação.
    - Ajustes de parâmetros de detecção e definição do número de quadros (32 frames) analisados por vídeo.

- **Desenvolvimento da Ferramenta**
  - Implementação de uma interface funcional para upload e análise de vídeos.
  - Integração do modelo de IA ao software, permitindo processamento local.
  - Teste de otimização de performance para diferentes níveis de hardware (testado em celulares e computadores com diferentes capacidades).

- **Preparação e Organização dos Dados de Teste**
  - Coleta de vídeos reais e manipulados disponíveis publicamente na internet.
  - Organização dos vídeos em categorias de resolução (360p e 1080p) e diferentes condições de iluminação e compressão.
  - Estruturação dos testes para comparação entre dispositivos e cenários variados.

- **Execução dos Experimentos**
  - Análise automática de quadros por vídeo.
  - Coleta de métricas: *score* de autenticidade, tempo de processamento, taxa de acerto e consistência entre dispositivos.
  - Comparação entre resultados de vídeos reais e falsos em diferentes resoluções e condições.

- **Avaliação dos Resultado**
  - Interpretação dos scores gerados pelo modelo e análise do comportamento da inferência.
  - Identificação de limitações: impacto da compressão, iluminação e *hardware*.
  - Verificação da capacidade de generalização do modelo para vídeos reais da internet, fora do *dataset* original de treinamento.

- **Conclusão e Propostas Futuras**
  - Consolidação dos resultados obtidos e validação da eficácia do sistema.
  - Sugestões de aprimoramento para detecção em tempo real, uso de outras arquiteturas e criação de base de dados própria para o modelo.

4. TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns trabalhos relacionados ao tema principal deste trabalho. Cada um deles evidência, a necessidade de métodos confiáveis de detecção, capazes de lidar com diferentes tipos de manipulações e variações presentes nas imagens.

4.1 Deepfake Detection Challenge (DFDC)

O *Deepfake Detection Challenge* (DFDC), impulsionado pela colaboração entre o *Facebook AI, Microsoft, Amazon Web Services* (AWS) e diversas universidades, surgiu para estimular a criação de ferramentas automatizadas capazes de identificar materiais alterados por inteligência artificial — com foco em vídeos *deepfake*.

Iniciado em 2019 e concluído em 2020, o desafio disponibilizou um conjunto de dados de cerca de 470 GB, incluindo mais de 100.000 vídeos autênticos e manipulados. Esses vídeos foram produzidos a partir de filmagens originais de atores, incorporando variações realistas de iluminação, posição, sexo, etnia, compressão e ruído, tornando-se um dos maiores e mais variados conjuntos de dados públicos já criados para este fim (DOLHARE et al., 2020).

Mais de 2.000 grupos participaram, explorando diversas estratégias como redes neurais convolucionais (CNNs), aprendizado por transferência, combinação de modelos e técnicas de tratamento prévio de imagens faciais. Contudo, o resultado do modelo campeão (da equipe Selim Seferbekov) revelou as dificuldades da tarefa: a pontuação mais alta foi de apenas 65,18% de AUC (Área sob a Curva) em vídeos não vistos antes, mostrando que os modelos, apesar de bons nos dados de treino e teste, ainda tinham dificuldade em se adaptar a situações do mundo real.

O DFDC teve um efeito significativo na comunidade científica, destacando a dificuldade de detectar *deepfakes* em cenários reais e com manipulações cada vez mais avançadas. Adicionalmente, estabeleceu referências e critérios para pesquisas futuras, e incentivou a criação de novos conjuntos de dados, estruturas de modelos e abordagens de avaliação (DOLHARE et al., 2020).

4.2 MesoNet

O *MesoNet*, proposto por Afchar et al. (2018), foi uma das primeiras arquiteturas de redes neurais projetadas especificamente para a detecção de *deepfakes* em vídeos e imagens. O nome “meso” vem da rede desenvolvida para trabalhar com características de nível intermediário (mesoscópico), que não precisam de detalhes muito finos nem de interpretações muito abstratas (AFCHAR et al., 2018).

A teoria surgiu como uma alternativa leve e eficaz, capaz de ser treinada rapidamente e com baixo custo computacional. A arquitetura é composta por apenas algumas camadas densas, o que a torna mais adequada para aplicações em tempo real ou em dispositivos de capacidade limitada.

O *MesoNet* apresentou bons resultados na detecção de manipulações em vídeos como *Deepfake* e Face2Face, mesmo quando os vídeos estavam comprimidos. Além disso, a simplicidade da estrutura permitiu maior interpretação dos resultados, mostrando quais regiões da face apresentavam padrões anômalos (AFCHAR et al., 2018).

Infelizmente, o *MesoNet* não foi capaz de alcançar a precisão de outros modelos mais profundos, como a *EfficientNet* ou *Xception*, em bases de dados mais complexas. Porém, o *MesoNet* ainda é considerado uma excelente escolha para aplicações rápidas e cenários com recursos computacionais limitados (AFCHAR et al., 2018).

4.3 FaceForensics++

O *FaceForensics++* é um dos benchmarks mais relevantes e amplos utilizados quando o assunto é detecção de *deepfakes*. Ele foi desenvolvido por Rössler et al. (2019) com o objetivo de fornecer uma base de dados padronizada e realista para

avaliar algoritmos que detectam manipulações faciais em vídeos (RÖSSLER et al., 2019).

A base de dados contém mais de 1.000 vídeos originais e suas respectivas versões manipulados com quatro técnicas populares:

- *DeepFakes*
- *Face2Face*
- *FaceSwap*
- *NeuralTextures*

Essas técnicas aplicam diferentes tipos de abordagens de manipulação facial, como substituição de rosto, reenquadramento de expressões e ajustes de textura, permitindo que os pesquisadores testem a capacidade dos modelos em diferentes cenários (RÖSSLER et al., 2019).

Um dos diferenciais do *FaceForensics*++ é que os vídeos são disponibilizados em diversos níveis de compressão, simulando ambientes reais, onde os vídeos passam por perda de qualidade. Isso permite fazer com que o modelo aprenda e teste modelos com condições próximas das que ocorrem fora do laboratório (RÖSSLER et al., 2019).

O benchmark também inclui diversas métricas padronizadas para comparar o desempenho entre diversos métodos, além de scripts de pré-processamento e segmentação facial (RÖSSLER et al., 2019).

4.4 Celeb-DF (Kaggle)

A Celeb-DF (*Celeb-Deepfake Detection Dataset*), também disponível no Kaggle, é bastante usada em estudos sobre a identificação de *deepfakes*. Yuezun Li et al. (2020), que propuseram essa coleção, tinham como objetivo ultrapassar as falhas dos bancos de dados antigos, como a baixa qualidade das imagens e a pouca variedade nos cenários de manipulação (LI; YANG; SUN; QI; LYU, 2019).

O *dataset* contém mais de 5.600 vídeos verdadeiros e cerca de 5.000 alterados com técnicas avançadas de *deepfake*. Com vídeos em diversos ambientes, incluindo variação na iluminação, ângulo, compressão e expressões faciais, de modo a simular condições mais próximas da vida real, como em redes sociais e plataformas de vídeo

(LI; YANG; SUN; QI; LYU, 2019; YUEZUN LI, 2020).

O Celeb-DF v2 se destaca pela melhoria na qualidade em comparação às versões anteriores e a outros bancos públicos. Os vídeos manipulados foram produzidos com algoritmos mais sofisticados, reduzindo os artefatos visuais que antes eram facilmente detectados por sistemas automáticos. Isso torna o desafio mais realista e contribui para a criação de modelos mais robustos (VARGHESE, 2022).

O Celeb-DF v2 no Kaggle facilita o acesso para a comunidade científica, pois não apenas garante os dados, mas também oferece códigos e exemplos de aplicação. Dessa forma, esse conjunto de dados é amplamente utilizado em trabalhos acadêmicos e competições de machine learning (YUEZUN LI; YANG; SUN; QI; LYU, 2019; VARGHESE, 2022; YUEZUN LI, 2020).

5. IMPLEMENTAÇÃO DO PROJETO

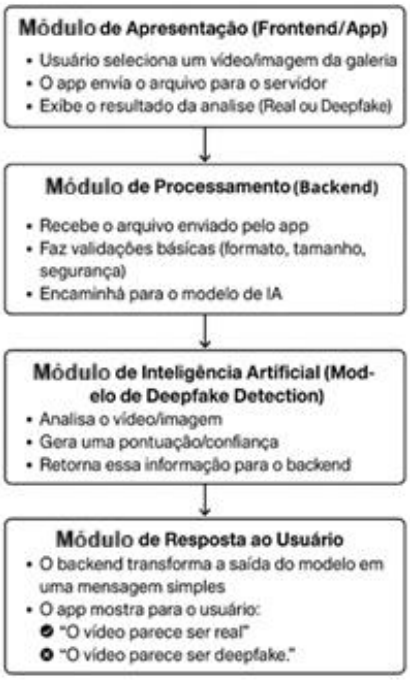
A implementação do sistema teve início com a utilização de um modelo previamente disponibilizado no repositório “selimsef/dfdc_deepfake_challenge”. Essa decisão foi motivada pela indisponibilidade de hardware adequado para treinar um modelo do zero, uma vez que o processo de treinamento exige GPUs de alto desempenho e uma infraestrutura computacional bastante avançada. Dessa forma, optou-se por adaptar um modelo já consolidado, permitindo focar a pesquisa na conversão, integração e adaptação do modelo para dispositivos móveis, além de otimizar o desempenho e a precisão na detecção de *deepfakes*.

5.1 Arquitetura do Projeto

O sistema proposto foi estruturado de forma funcional, com módulos que trabalham em conjunto para realizar a análise de vídeos e identificar possíveis manipulações faciais. Cada módulo desempenha um papel essencial dentro do fluxo de operação, como mostra a Figura 3.

- **Modulo de Apresentação (*Front-end*)**
- **Modulo de Processamento (*Back-end*)**
- **Modulo de Inteligência Artificial (Modelo de *Deepfake*)**
- **Modulo de Resposta ao Usuário**

Figura 3 - Camadas de Apresentação



Fonte: Autores (2025)

5.2 Arquitetura do Software

Conforme explicado anteriormente, a arquitetura do sistema proposto é composta por quatro camadas principais: Camada de Apresentação, Camada de Processamento, Camada de Inteligência Artificial e uma Camada de resposta ao Usuário. Neste tópico, vamos detalhar cada uma dessas camadas, bem como as tecnologias e ferramentas utilizadas.

5.2.1 Módulo de Apresentação (Front-end)

A Camada de Apresentação corresponde à interface do sistema com a qual o usuário interage diretamente. Seu objetivo é fornecer uma experiência intuitiva, objetiva e de fácil navegação, permitindo que qualquer usuário consiga realizar a análise de um vídeo com o mínimo de etapas possíveis.

Para o desenvolvimento dessa camada, utilizou-se o *Android Studio*, com a linguagem *Kotlin*, que permitiu portar a aplicação para dispositivos Android e integrar a interface com os componentes lógicos responsáveis por processar os vídeos. A escolha dessa ferramenta garantiu compatibilidade com diferentes versões do sistema

operacional, além de facilitar a integração com bibliotecas nativas de acesso à galeria e renderização de elementos visuais.

O *layout* foi planejado com foco na usabilidade e simplicidade, evitando excesso de elementos visuais e destacando apenas o que é essencial para a tarefa principal: selecionar um vídeo e receber o resultado da análise. Os elementos foram organizados com base no conceito de ação direta, destacando um botão principal para upload e uma área de retorno visual para exibição do resultado.

As principais funcionalidades implementadas nessa camada incluem:

- Seleção de vídeos diretamente da galeria do dispositivo.
- Envio do vídeo para a camada de processamento, com indicação visual de carregamento.
- Exibição do score de autenticidade com uma resposta clara, indicando se o vídeo é classificado como “Real” ou contém manipulação (*Deepfake*).
- Feedback visual rápido e direto, reduzindo a necessidade de interpretação técnica por parte do usuário.

5.2.2 Módulo de Processamento (Back-end)

A Camada de Processamento, ou *Back-end*, é responsável por gerenciar toda a lógica de negócio, realizar validações, processar os dados recebidos e estabelecer a comunicação com o modelo de detecção de *deepfakes*. Essa camada atua como um intermediário eficiente entre a interface do usuário e a camada de inteligência artificial, garantindo que as requisições sejam tratadas de forma segura, rápida e escalável.

Para o desenvolvimento do *Back-end* do sistema de detecção de *deepfakes*, foram adotadas tecnologias voltadas tanto para o processamento eficiente de vídeos quanto para a inferência de modelos de inteligência artificial.

O sistema foi construído utilizando Python, e o modelo de detecção foi desenvolvido e treinado com *PyTorch*.

Durante o processamento, ferramentas como *NumPy* foram utilizadas para manipulação de matrizes e estruturas numéricas. Para o tratamento e análise dos vídeos, foram utilizadas as bibliotecas *OpenCV* e *FFmpeg*, que permitem a extração de frames e a conversão dos formatos de mídia, responsáveis por redimensionar o arquivo a ser analisada pela IA para o formato 380x380.

Para garantir portabilidade e padronização do ambiente de execução, o *Back-end* foi containerizado utilizando *Docker*, para que funcione da mesma forma que o projeto original. A detecção facial foi otimizada com o uso da MTCNN *(Multi-task Cascaded Convolutional Networks*), responsável por localizar e recortar rostos de forma precisa, permitindo que o modelo de IA foque apenas nas regiões relevantes, que posteriormente com o porte para o aplicativo, foi utilizado o ML KIT por conta da compatibilidade com o Android, já que o MTCNN não está disponível para esse tipo de sistema.

Todas essas recomendações foram seguidas utilizando como base o projeto do autor para o desafio do Kaggler.

Posteriormente todo o projeto foi convertido para uso em dispositivos Android utilizando *PyTorch Mobile*, permitindo a usuabilidade com o ecossistema Android. E como dito anteriormente, passamos a usar o ML Kit ao invés do MTCNN, que é um SDK para dispositivos Android, para suportar as funções necessárias para a IA funcionar corretamente.

As principais funções dessa camada incluem:

- Receber os vídeos enviados pelo *front-end*, garantindo que estejam em formatos suportados e dentro do limite de tamanho permitido.
- Gerenciar as requisições de análise, enviando os vídeos para a camada de inteligência artificial.
- Monitorar o status das análises em tempo real e retornar os resultados ao frontend de forma eficiente.

5.2.3 Módulo de Inteligência Artificial (Modelo de Deepfake)

A camada de inteligência artificial constitui o núcleo técnico do sistema, sendo responsável pela análise dos vídeos enviados e pela detecção de possíveis manipulações de *deepfake*. Essa análise é realizada por meio de modelos de *Deep Learning* (aprendizado profundo) baseados em redes neurais convolucionais (CNNs), especificamente a arquitetura *EfficientNet*, que são altamente eficazes na identificação de padrões complexos presentes em imagens e vídeos.

O modelo foi pré-treinado com *ImageNet* (*Dataset*) e melhorado com o método *Noisy Student* (Tecnica de Autotreinamento), o que o tornou um classificador de imagens excepcionalmente preciso e robusto. Com isso temos uma abordagem de

Transfer Learning (Aprendizado por Transferência), onde um modelo treinado em uma tarefa (no caso o modelo original com o ImageNet) é ajustado para uma tarefa específica (detecção de *deepfakes*).

A IA usa uma abordagem de classificação quadro a quadro (*frame-by-frame classification*). Isso significa que a CNN processa cada quadro de vídeo individualmente para determinar se contém uma *deepfake. E*les são divididos em 32 quadros de forma aleatória, o que permite que o modelo faça uma previsão dos vídeos de forma rápida o suficiente para ser viável. E não analisando todos frames do vídeo, o que seria custoso demais em tempo e processamento.

Para o Pré-processamento e detecção de rosto é utilizada a biblioteca MTCNN (*Multi-task Cascaded Convolutional Networks*), que é uma CNN especializada para as tarefas de detecção de rosto e alinhamento (*face alignment*), que no porte para Android foi substituída para o ML Kit, por conta da compatibilidade com o Android.

Ou seja, para maximizar a robustez e a precisão do sistema de classificação de *deepfakes*, adotou-se a técnica de Aprendizado por Conjunto (*Ensemble Learning*) utilizando cinco modelos treinados. Esta abordagem visa mitigar os erros individuais de um único modelo, utilizando o princípio do consenso entre múltiplos especialistas. Posteriormente isso no nosso projeto foi descartado e utilizamos apenas um modelo.

Principais funções dessa camada:

- Receber vídeos enviados pelo *Front-end*.
- Realizar pré-processamento dos vídeos, incluindo extração de frames, redimensionamento e normalização.
- Comunicar-se com o modelo de IA para inferência.
- Retornar os resultados ao *Front-end*, indicando se o vídeo é “Real” ou “*Deepfake*”.

5.2.4 Módulo de Resposta ao Usuário

Por fim, esta camada é responsável por consolidar e apresentar os resultados ao usuário de forma clara, objetiva e acessível.

Suas principais funções incluem:

- Receber os resultados processados pela camada de inteligência artificial.
- Transformar a saída do modelo em mensagens simples, como “*Real*” ou “*Deepfake*”.

- Exibir o resultado na interface do usuário, junto com a imagem ou o vídeo analisado.
- Fornecer feedback visual ao usuário, indicando o status da análise e os resultados obtidos de forma compreensível.

Essa camada garante que o usuário receba uma resposta rápida e clara, promovendo uma experiência de uso satisfatória e confiável.

5.3 Implementação do Software

5.3.1 Análise de requisitos

Com o intuito de demonstrar o funcionamento do sistema, os tópicos a seguir mostram os requisitos funcionais e não funcionais do protótipo.

5.3.1.1 Requisitos funcionais

Os requisitos funcionais referem-se às capacidades ou serviços que um sistema deve oferecer, e sua definição depende de vários aspectos, como a categoria de *software* utilizada, a natureza do sistema a ser criado e as expectativas dos usuários. Embora todos os exemplos mencionados sejam considerados requisitos funcionais, a complexidade desses requisitos varia conforme o público a quem se destina a descrição; para os usuários, eles são expressos de maneira mais geral, enquanto para o sistema, as funções são elaboradas com maior detalhamento.

Os requisitos funcionais do nosso programa são:

Para melhor observar ambos os requisitos funcionais, eles foram reunidos na Tabela 1:

Tabela 1 - Requisitos Funcionais

Relação de Requisitos Funcionais	
ID	**Descrição**
RF01	O sistema fornece ao usuário uma tela para entrada do vídeo que se deseja analisar, com um botão para se carregar o vídeo.
RF02	O sistema apresenta uma tela de gerenciador de arquivos no disco para que o usuário escolha a o vídeo a ser analisado.
RF03	Após analisado do vídeo, o sistema apresenta ao usuário junto a uma porcentagem de confiabilidade da IA em sua análise, o tempo de processamento, e o resultado confirmando ou não o uso de *deepfake* no vídeo.
RF04	Caso não se encontre uma face humana no vídeo, o sistema apresenta uma mensagem de erro informando ao usuário que não foi possível encontrá-lo.

Fonte: Autores (2025)

5.3.1.2 Requisitos não funcionais

Os requisitos não funcionais se referem a aspectos do sistema como um todo, estabelecendo suas características e limitações, como a confiabilidade e a capacidade de armazenamento. Alguns desses requisitos podem até influenciar o método que será adotado na criação do sistema, como a escolha de uma linguagem de programação específica.

Esses requisitos não funcionais podem ser mais importantes do que os funcionais, pois determinam aspectos como qualidade, segurança, velocidade e confiança do sistema. Quando esses requisitos não são cumpridos, isso pode levar a falhas graves em um sistema, mesmo que suas funções estejam corretamente implementadas, podendo torná-lo assim ineficaz.

Para melhor observar ambos os requisitos não funcionais, eles foram reunidos na tabela 2:

Tabela 2 - Requisitos não Funcionais

Relação de Requisitos Não Funcionais	
ID	**Descrição**
RNF01	Aplicativo para sistemas *Android*
RNF02	Taxa de confiabilidade deve ser >0,5 para afirmação de imagem falsa
RNF03	Sistema analisa apenas um vídeo por vez

Fonte: Autores (2025)

5.3.2 Descrição dos Casos de Uso

Os casos de uso representam, de forma simplificada, como o sistema se comporta sob a perspectiva do usuário. Eles descrevem as ações possíveis, os atores envolvidos e as interações que ocorrem no processo. No contexto deste projeto, o ator principal é o usuário, que utiliza o aplicativo para enviar vídeos e receber o resultado da análise, indicando se o conteúdo é real ou *deepfake*.

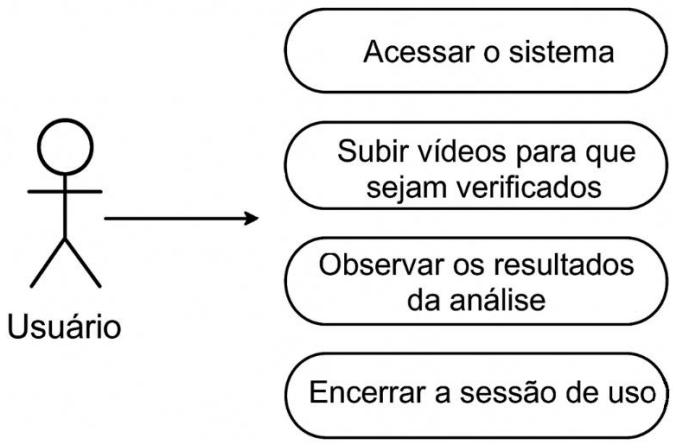
As principais ações disponíveis para o usuário são:

- Acessar o sistema
- Subir vídeos para que sejam verificados
- Observar os resultados da análise, com destaque para a confiança do sistema no diagnóstico
- Encerrar a sessão de uso

Cada funcionalidade é considerada essencial para garantir a praticidade e a eficiência do sistema proposto.

A Figura 4, apresenta o Diagrama de Casos de Uso que ilustra as interações do usuário com o sistema:

Figura 4- Casos de Uso do Usuário



Fonte: Autores (2025)

A tabela descrevendo as ações do usuário se encontra na sessão de **APÊNDICES**.

5.3.3 Conversão e Preparação do Modelo

O modelo original, baseado na arquitetura *EfficientNet* B7, foi disponibilizado em implementações no *PyTorch* e *TensorFlow*. Para viabilizar sua execução no ambiente Android, foi necessário convertê-lo para o formato *TorchScript*, compatível com *PyTorch Mobile*.

Esse processo foi realizado de forma iterativa: inicialmente, diferentes tentativas de conversão apresentaram falhas, como ausência de funções de pré-processamento e parâmetros incorretos. Após ajustes sucessivos, obteve-se um script estável que carregava corretamente os pesos, garantindo a mesma acurácia observada no ambiente *Python*. O resultado foi a geração de arquivos .pt que podiam ser embarcados diretamente no aplicativo.

5.3.4 Pipeline de Pré-processamento

A detecção de *DeepFakes* exige que os quadros do vídeo sejam tratados de forma consistente, como vemos na figura 5. No ambiente *Python*, o *pipeline* utilizava

as seguintes etapas, como mostra a Figura 5:

1. Extração de quadros de cada vídeo.
2. Detecção do maior rosto com MTCNN.
3. Recorte do rosto com *padding* de 30% para preservar detalhes contextuais.
4. Centralização em um *canvas* preto 380×380 pixels, preservando a proporção.
5. Normalização dos pixels conforme os parâmetros utilizados no treinamento (*ImageNet*).

Figura 5 - *Pipeline* de Pré-processamento

```python
def predict_on_single_video(video_path, models, face_extractor, input_size=380, frames_per_video=32):
    print(f"\n🎬 Processing video: {os.path.basename(video_path)}")

    frames = face_extractor.video_read_fn(video_path)
    if frames is None:
        print("⚠️ Failed to read frames from video.")
        return 0.5

    face_dicts = face_extractor.process_video(video_path)

    scores = []
    for frame_idx, frame_data in enumerate(face_dicts):
        for f_idx, face in enumerate(frame_data["faces"]):
            if face is None:
                continue

            # Resize and center in black canvas
            face = isotropically_resize_image(face, input_size)
            face = put_to_center(face, input_size)

            # Convert to tensor
            with torch.no_grad():
                tensor = torch.from_numpy(face.transpose((2, 0, 1))).unsqueeze(0).float() / 255.0
                tensor = tensor.to("cuda")
                tensor = F.interpolate(tensor, size=(input_size, input_size), mode='bilinear', align_corners=False)

                # Normalize
                mean = torch.tensor([0.485, 0.456, 0.406], device="cuda").view(1, 3, 1, 1)
                std = torch.tensor([0.229, 0.224, 0.225], device="cuda").view(1, 3, 1, 1)
                tensor = (tensor - mean) / std

                tensor = tensor.half()

                total_score = 0.0
                for m_idx, model in enumerate(models):
                    output = model(tensor)
                    raw_score = output.item()
                    sigmoid_score = sigmoid(raw_score)
                    print(f"🧠 Frame {frame_idx:02d} Face {f_idx} - Model {m_idx}: raw {raw_score:.4f}, sigmoid {sigmoid_score:.4f}")
                    total_score += sigmoid_score

                avg_score = total_score / len(models)
                scores.append(avg_score)
                print(f"✅ Frame {frame_idx:02d} Face {f_idx} average score: {avg_score:.4f}")
```

Fonte: Autores (2025)

No ambiente Android, entretanto, a biblioteca MTCNN não estava disponível. Para contornar essa limitação, foi adotada a API de *Face Detection* do ML Kit (*Google*), que cumpriu a mesma função de identificar rostos nos quadros. Apesar de pequenas diferenças nos *bounding boxes* detectados, buscou-se replicar o mesmo *pipeline*: aplicação de *padding*, centralização em *canvas* preto e normalização padrão. Essa adaptação permitiu aproximar ao máximo os resultados entre os dois ambientes.

5.3.5 Ensemble de Modelos

Inicialmente, o aplicativo foi desenvolvido para suportar um *ensemble* de três modelos *TorchScript*. Em cada vídeo analisado, os quadros eram processados pelos três modelos, cujas saídas (*sigmoid*) eram combinadas por média simples. Em seguida, aplicava-se a estratégia *confidentStrategy*, que agrega os resultados de

múltiplos quadros de forma robusta, reduzindo a variabilidade entre predições individuais.

No entanto, após testes comparativos, verificou-se que o desempenho de um único modelo não apenas se igualava ao *ensemble* de três modelos, como também proporcionava maior eficiência e rapidez na execução em dispositivos móveis. Assim, a versão final do aplicativo foi simplificada para utilizar apenas um modelo *TorchScript*, garantindo resultados equivalentes com menor custo computacional.

5.3.6 Implementação no Android

O aplicativo foi desenvolvido em *Kotlin* no Android Studio, com suporte ao *PyTorch* Mobile e ao ML Kit. Os principais ajustes realizados no *MainActivity* incluem:

- Interface do Usuário: botões para seleção de vídeo e imagem, barra de progresso durante o processamento e visualização de miniaturas dos rostos detectados. Como mostra a Figura 6.

Figura 6 - Implementação em Android

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    selectVideoButton = findViewById(R.id.selectVideoButton)
    progressBar = findViewById(R.id.progressBar)
    resultText = findViewById(R.id.resultText)
    faceContainer = findViewById(R.id.faceContainer)
    faceScroll = findViewById(R.id.faceScroll)

    // Detector será criado fresco por vídeo (sem tracking)
    loadModels()

    selectVideoButton.setOnClickListener {
        pickVideo.launch("video/*")
    }
}
```

Fonte: Autores (2025)

- Carregamento do Modelo: cópia automática dos arquivos .pt para o diretório interno do app, validação de integridade e execução de *warmup* para garantir estabilidade antes da primeira inferência. Como mostra a Figura 7.

Figura 7 - Carregamento do modelo

```
private fun loadModels() {
    try {
        val path = assetFilePath("555_19.pt")
        val sizeMb = File(path).length().toDouble() / (1024.0 * 1024.0)
        Toast.makeText(this, String.format("Model asset: %.1f MB", sizeMb), Toast.LENGTH_SHORT).show()

        val m = Module.load(path)

        // Warmup 1×3×380×380
        val zeros = Tensor.fromBlob(
            FloatArray(1 * 3 * 380 * 380),
            longArrayOf(1, 3, 380, 380)
        )
        val warm = m.forward(IValue.from(zeros)).toTensor().dataAsFloatArray
        val w0 = warm.firstOrNull() ?: Float.NaN
        Toast.makeText(this, "Warmup out[0]: $w0", Toast.LENGTH_SHORT).show()

        modules = listOf(m)
    } catch (t: Throwable) {
        modules = emptyList()
        Toast.makeText(this, "Failed to load model: ${t.message}", Toast.LENGTH_LONG).show()
    }
}
```

Fonte: Autores (2025)

- Pré-processamento: recorte com *padding* de 30%, centralização em canvas 380×380 e normalização padrão, assegurando consistência com o ambiente *Python*. Como mostra a Figura 8.

Figura 8 - Pré-processamento

```
private fun processVideo(uri: Uri) {
    progressBar.visibility = View.VISIBLE
    resultText.text = ""
    faceContainer.removeAllViews()
    faceScroll.visibility = View.GONE

    val startNs = System.nanoTime()

    ioScope.launch {
        // Detector fresco para ESTE vídeo (sem estado entre vídeos)
        val detector = createDetector()

        // Coletor de probabilidades por modelo (streaming, sem guardar bitmaps)
        val perModelProbs: Array<MutableList<Double>> =
            Array(modules.size) { mutableListOf<Double>() }

        // Guardar somente 24 thumbs para UI
        val thumbs: MutableList<Bitmap> = mutableListOf()
        val maxThumbs = 24

        // Teto de faces para inferência (limita memória/tempo)
        val maxFaces = 32 * 4
        var facesCount = 0

        // Extração de frames (aqui ainda retorna lista, mas reciclamos cada frame após uso)
        val frames = extractUniformFrames(uri, framesPerVideo = 32)
```

Fonte: Autores (2025)

- Inferência: execução do modelo em lote com tratamento de exceções, descartando valores inválidos e assegurando retorno confiável (probabilidade ∈ [0,1]), Como mostra as Figuras 9 e 10.

Figura 9 – Inferência 1

```
private fun inferOne(m: Module, bmp380: Bitmap): Double {
    // Preenche buffer com a imagem normalizada
    inputBuffer380.rewind()
    TensorImageUtils.bitmapToFloatBuffer(
        bmp380, 0, 0, 380, 380,
        floatArrayOf(0.485f, 0.456f, 0.406f),
        floatArrayOf(0.229f, 0.224f, 0.225f),
        inputBuffer380, 0
    )
    val input = Tensor.fromBlob(inputBuffer380, longArrayOf(1, 3, 380, 380))

    val out: Tensor = try {
        m.forward(IValue.from(input)).toTensor()
    } catch (_: Throwable) {
        return Double.NaN
    }

    val scores = out.dataAsFloatArray
    if (scores.isEmpty()) return Double.NaN
```

Fonte: Autores (2025)

Figura 10 - Inferência 2

```
    // Mapeia para probabilidade de FAKE
    return when (scores.size) {
        1 -> {
            val v = scores[0].toDouble()
            if (v in 0.0 ≤ .. ≤ 1.0) v else 1.0 / (1.0 + kotlin.math.exp(-v))
        }
        2 -> {
            val s = scores[0] + scores[1]
            val looksSoftmax = s > 0.98f && s < 1.02f && scores[0] >= 0f && scores[1] >= 0f
            if (looksSoftmax) {
                scores[fakeIndexHint].toDouble()
            } else {
                1.0 / (1.0 + kotlin.math.exp(-scores[fakeIndexHint].toDouble()))
            }
        }
        else -> {
            val v = scores[0].toDouble()
            if (v in 0.0 ≤ .. ≤ 1.0) v else 1.0 / (1.0 + kotlin.math.exp(-v))
        }
    }
}
```

Fonte: Autores (2025)

- Agregação Temporal: aplicação da *confidentStrategy*, capaz de suavizar as predições ao longo dos quadros, Como mostra a Figura 11.

Figura 11 - Agregação Temporal

```
private fun confidentStrategy(pred: List<Double>, t: Double = 0.8): Double {
    if (pred.isEmpty()) return 0.5
    val sz = pred.size
    val fakes = pred.count { it > t }
    return when {
        (fakes > sz / 2.5) && (fakes > 11) -> pred.filter { it > t }.average()
        pred.count { it < 0.2 } > 0.9 * sz  -> pred.filter { it < 0.2 }.average()
        else -> pred.average()
    }
}
```

Fonte: Autores (2025)

- Experiência do Usuário: mensagens claras em casos de erro (“*No face found*…”), transparência com miniaturas dos rostos processados e *feedback* visual contínuo para não travar a interface, Como mostra a Figura 12.

Figura 12 - Experiência do Usuário

```
            withContext(Dispatchers.Main) {
                val elapsedSec = (System.nanoTime() - startNs) / 1_000_000_000.0
                progressBar.visibility = View.GONE

                if (facesCount == 0) {
                    resultText.text = "No face found in sampled frames."
                } else {
                    faceScroll.visibility = View.VISIBLE
                    // Exibe somente as thumbs coletadas
                    for (c in thumbs) addFaceThumb(c)
                    val label = if (score >= 0.5) "FAKE" else "REAL"
                    resultText.text = String.format("Score: %.4f  →  %s\nTempo total: %.3f s", score, label, elapsedSec)
                }
            }
        }
```

Fonte: Autores (2025)

5.3.7 Limitações Identificadas

Apesar da adaptação bem-sucedida, algumas diferenças foram observadas entre o ambiente Python e o aplicativo Android. A principal delas refere-se ao detector de rostos: enquanto no Python utilizou-se o MTCNN, no Android adotou-se o ML Kit, resultando em pequenas variações nos recortes faciais. Essas diferenças impactaram levemente a consistência das pontuações finais, e diminui a acurácia geral.

Esse ponto foi registrado como melhoria futura, visando aproximar ainda mais os *pipelines* entre desktop e *mobile*.

5.4 Funcionamento e Interfaces

O sistema desenvolvido tem como objetivo a identificação de vídeos manipulados (*deepfakes*) a partir do envio de arquivos pelo usuário. Nessa sessão será apresentado detalhadamente as interfaces do sistema e suas funcionalidades. Seu funcionamento segue um fluxo simples e intuitivo:

5.4.1 Acesso ao sistema

A Figura 13 apresenta a tela de carregamento do aplicativo, exibida imediatamente após sua abertura. Nessa etapa, o sistema realiza a inicialização dos componentes internos e o carregamento do modelo de detecção de *DeepFakes* em PyTorch Mobile para que esteja disponível na memória. Durante esse processo, é exibida uma tela simples de *loading*, que tem como objetivo indicar ao usuário que a aplicação está sendo preparada para uso.

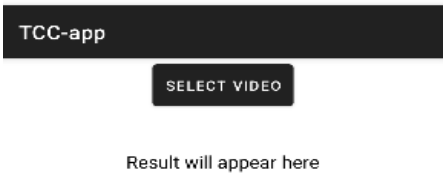
Figura 13 - Tela de Carregamento do Aplicativo

Fonte: Autores (2025)

5.4.2 Submissão de vídeos

O usuário seleciona e envia o arquivo de vídeo desejado armazenados na galeria do dispositivo, através do botão “*Select Vídeo*”, que é encaminhado para o módulo de análise, como podemos ver na figura 14.

Figura 14 - Tela Inicial do Aplicativo

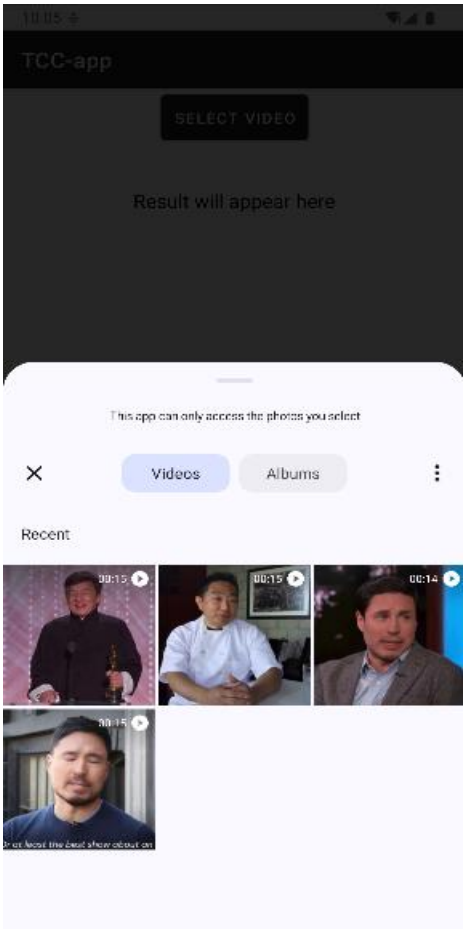


Fonte: Autores (2025)

5.4.3 Processamento pela IA

A Figura 15 mostra a etapa de seleção de vídeo. Após clicar no botão "*Select Video*", o usuário escolhe um arquivo de mídia para ser analisado pelo sistema. Uma vez carregado o vídeo, o aplicativo inicia a extração de quadros e a detecção de rostos.

Figura 15 - Etapa de Seleção de Vídeo

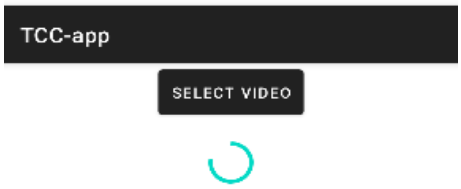


Fonte: Autores (2025)

5.4.4 Tela de Carregamento de Vídeo

A Figura 16 exibe uma barra de progresso que acompanha a execução do processamento, oferecendo ao usuário um indicativo visual de que a análise está em andamento.

Figura 16 - Tela de Carregamento de Vídeo

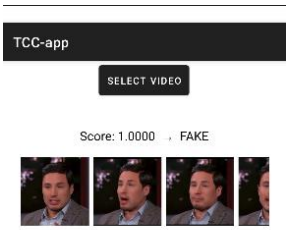


Fonte: Autores (2025)

5.4.5 Exibição dos resultados

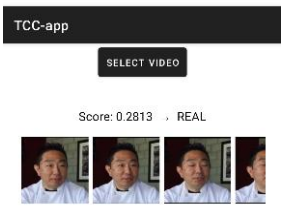
O sistema retorna o diagnóstico através de um texto, indicando se o vídeo é autêntico (*Real*) ou manipulado (*Fake*), acompanhado de um percentual de confiança (*Score*), que representa o nível de confiança da predição e apresenta a área de miniaturas de rostos detectados. Que facilitam a interpretação, como podemos ver nas figuras 17 e 18.

Figura 17 - Exibição de Resultados FAKE



Fonte: Autores (2025)

Figura 18 - Exibição de Resultados REAL

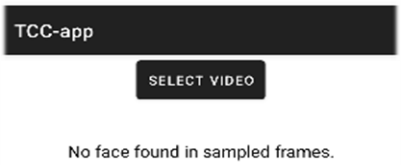


Fonte: Autores (2025)

5.4.6 Mensagem de Erro

Por fim, a Figura 19 exemplifica as mensagens de *feedback* e erros. Em situações em que não há detecção de rosto ou quando ocorre algum problema durante o processamento, o aplicativo exibe mensagens como "*No face found*" ou notificações de erro adequadas, garantindo clareza e boa experiência.

Figura 19 - Mensagem de Erro



Fonte: Autores (2025)

A interface foi projetada para ser simples, acessível e objetiva, permitindo que usuários com diferentes níveis de conhecimento técnico consigam utilizá-la. As principais telas apresentam:

- Botão de upload de vídeo;
- Barra de progresso durante o processamento;
- Área de resultados, com mensagem clara sobre a autenticidade do vídeo, percentual de confiança, tempo de processamento e elementos gráficos de apoio.

Essa abordagem garante que o usuário final tenha uma experiência fluida e confiável, ao mesmo tempo em que o sistema mantém a robustez técnica necessária para lidar com análises complexas de mídia digital.

6. TESTES E RESULTADOS

Esta seção é referente a execução dos testes feitos para validação do sistema de identificação de vídeos *deepfake*.

6.1 Testes Iniciais de validação do Sistema de identificação de vídeos gerados por IA.

Para validação do sistema de identificação de vídeos gerados por IA, diversos testes foram feitos no aplicativo utilizando o *dataset* Celeb-DF, devido à falta de acesso ao *dataset* original. Descritos a seguir os testes conduzidos:

6.1.1 Validação do modelo original

O primeiro teste foi feito executando um arquivo *python* já incluso no material original, em uma imagem Docker cedida pelo autor, que apresentava em terminal os valores de predição do modelo de cada vídeo apontado em uma pasta. Fazendo assim uma validação de que cada modelo estava com pesos definidos corretamente e suas acurácias dentro do esperado pelo valor apresentado pelo autor. O teste ocorreu sem problemas e os valores obtidos estavam dentro do esperado.

6.1.2 – Conversão com PyTorch

O segundo teste foi feito com os modelos após a conversão feita com o *PyTorch*. Porém, foram usados parâmetros incorretos e estavam faltando algumas funções cruciais de pré-processamento de imagem. Devido a isso, o resultado obtido foi valores de acurácia sempre perto do 50%, dando a entender total aleatoriedade ao invés de predições precisas. Como mostra a tabela 3.

Tabela 3 - Resultados teste 2

MODELO:	ACC
111 36	0.46
555 19	0.42
777 29	0.38
777 31	0.44
888 37	0.44
888 40	0.49
999 23	0.43

Fonte: Autores (2025)

6.1.3 Ajuste nos parâmetros de conversão

Após ajustes serem feitos nos parâmetros de conversão para melhor se aproximar daquele usado no modelo original, a fim de manter o mais fiel possível os modelos ao original, para se obter resultados semelhantes. Também foram feitos ajustes no código do app para ler e utilizar corretamente os novos parâmetros. Porém mesmo com essas mudanças ainda não houve resultados positivos, pois ainda faltavam algumas funções de pré-tratamento de imagem necessárias, obtendo assim novamente resultados de predições próximos ao 50% (Total aleatoriedade). Como mostra a tabela 4.

Tabela 4 - Resultados teste 3

MODELO:	ACC
111 36	0.47
555 19	0.49
777 29	0.49
777 31	0.47
888 37	0.54
888 40	0.53
999 23	0.5

Fonte: Autores (2025)

6.1.4 Versão final do script de conversão

Após a adição das funções de pré-processamento de imagem que faltavam no *script* de conversão, o modelo que passou pela conversão do *script* finalizado apresentou resultados promissores e satisfatórios. Como mostra a tabela 5.

Tabela 5 - Resultados teste 4

MODELO:	ACC
999 23	0.88333
888 40	0.933333
888 37	0.916667
777 31	0.9
777 29	0.85
555 19	0.933333
111 36	0.88333

Fonte: Autores (2025)

6.1.5 Ajuste de parâmetros e comparação de modelos

No quinto teste foram utilizados os três modelos de maior acurácia obtidos com o teste 4 (888 40, 888 37 e 555 19), e em um novo script foram testados como se comportavam separadamente, junto com um valor de previsão com ensemble dos 3 modelos, e após os resultados o script, por meio de uma varredura de valores a cada 0,05 de *threshold*, era sugerido um novo parâmetro a ser usado para se encontrar um valor ideal a ser utilizado.

Na primeira iteração do teste, foi utilizado o parâmetro de *threshold* em 1 como feito pelo autor do modelo original e foi sugerido pelo script o uso apenas do modelo 555 19, e tendo uma acurácia nos vídeos testados de 93,3%. A maior acurácia de 96% em 3 previsões no *sweep* do *threshold*: 0,1 - 0,15 - 0,2.

Na segunda iteração do teste, com o *threshold* em 0,1 como sugerido no teste anterior, foi obtido novamente o modelo 555 19 como melhor resultado, e uma acurácia nos vídeos testados de 93,3%. A maior acurácia em 95% em 5 previsões no *sweep* do *threshold*: 0,7 – 0,75 – 0,8 – 0,85 – 0,9 – 0,95.

A terceira iteração foi feita com 0,7 de *threshold* e apenas o modelo 555 19, por motivos de redução do tempo na execução do teste, e foi obtido uma acurácia nos vídeos testados de 95%. A maior acurácia de 95% em 5 previsões no *sweep* do *threshold*: 0,7 – 0,75 – 0,8 – 0,85 – 0,9 – 0,95

Sendo assim escolhido o modelo 555 19 com *threshold* de 0,7 para ser usado no aplicativo.

6.2 Testes Finais com o aplicativo

Neste tópico, serão apresentados os testes finais realizados com o aplicativo já em funcionamento. O objetivo é demonstrar como a ferramenta se comporta em condições reais de uso, avaliando seu desempenho, estabilidade e usabilidade. Esses testes permitem identificar possíveis ajustes, além de validar se o sistema atende às expectativas e requisitos definidos ao longo do desenvolvimento.

6.3 Teste 1 (Verificando Acurácia do aplicativo no Dataset CelebDF)

Testes foram realizados em vídeos de qualidade 360p do *dataset* CelebDF, usando diferentes aparelhos e plataformas. Ao todo, foram analisados 20 vídeos por aparelho, sendo 10 vídeos falsos (*Fake*) e 10 vídeos reais (*Real*). Podemos verificar através das tabelas 6, 7, 8, 9, 10 e 11.

Tabela 6 - Teste 1 Redmi 9

Tempo de processamento		**Score**		**Acurácia**	
Mínimo:	139,77 s	Variação Fake:	0,9618 e 0,9932	Acurácia vídeo Fake:	100%
Máximo:	186,34 s	Variação Real:	0,0145 e 0,2581	Acurácia vídeos Real:	100%
Média:	157,27 s	Média geral (Fake):	~0,9815	Acurácia global:	100%
Mediana:	157,18 s	Média geral (Real):	~0,0689		
Desvio-padrão:	~14,5 s				

Fonte: Autores (2025)

Tabela 7 - Teste 1 Poco X5 Pro

Tempo de processamento		**Score**		**Acurácia**	
Mínimo:	41,50 s	Variação Fake:	0,4522 e 0,9901	Acurácia vídeo Fake:	90%
Máximo:	52,92 s	Variação Real:	0,0111 e 0,9771	Acurácia vídeos Real:	80%
Média:	43,46 s	Média geral (Fake):	~0,9306	Acurácia global:	85%
Mediana:	42,99 s	Média geral (Real):	~0,3043		
Desvio-padrão:	~2,32 s				

Fonte: Autores (2025)

Tabela 8 -Teste 1 Galaxy A51

Tempo de processamento		**Score**		**Acurácia**	
Mínimo:	101,51 s	Variação Fake:	0,1693 e 0,9922	Acurácia vídeo Fake:	91%
Máximo:	199,63 s	Variação Real:	0,0014 e 0,9591	Acurácia vídeos Real:	91%
Média:	140,81 s	Média geral (Fake):	~0,9141	Acurácia global:	91%
Mediana:	138,84 s	Média geral (Real):	~0,1252		
Desvio-padrão:	~16,43 s				

Fonte: Autores (2025)

Tabela 9 - Teste 1 Pc 1 (Ryzen 5 5500G – 16 GB RAM)

Tempo de processamento		**Score**		**Acurácia**	
Mínimo:	101,51s	Variação Fake:	0, 9955	Acurácia vídeo Fake:	90,90%
Máximo:	186,48 s	Variação Real:	0,1403	Acurácia vídeos Real:	90,90%
Média:	148,17 s	Média geral (Fake):	~0,9654	Acurácia global:	90,90%
Mediana:	146,84 s	Média geral (Real):	~0,2581		
Desvio-padrão:	~21,95 s				

Fonte: Autores (2025)

Tabela 10 - Teste 1 Galaxy A14

Tempo de processamento		**Score**		**Acurácia**	
Mínimo:	32,68s	Variação Fake:	0,2395 – 0,9999	Acurácia vídeo Fake:	53,85%
Máximo:	67,17s	Variação Real:	0,2844 – 0,9811	Acurácia vídeos Real:	57,14%
Média:	51,25s	Média geral (Fake):	~0,7090	Acurácia global:	55%
Mediana:	51,32s	Média geral (Real):	~0,5891		
Desvio-padrão:	~8,95s				

Fonte: Autores (2025)

Tabela 11 - Teste 1 Motorola G54

Tempo de processamento		**Score**		**Acurácia**	
Mínimo:	88.598s	Variação Fake:	0,0458 – 0,2074	Acurácia vídeo Fake:	90%
Máximo:	98.128s	Variação Real:	0,3482 – 0,4547	Acurácia vídeos Real:	90%
Média:	91.186s	Média geral (Fake):	~0,0207	Acurácia global:	90%
Mediana:	89.991s	Média geral (Real):	~0,4547		
Desvio-padrão:	~3.39s				

Fonte: Autores (2025)

No caso do PC2, os testes feitos anteriormente durante o desenvolvimento do aplicativo tinham apontado um resultado positivo e foram descritos no Teste 4.

6.4 Teste 2 (Comparativo de qualidade de vídeo)

O objetivo deste teste é analisar a acurácia, o tempo de processamento e a capacidade de detecção de adulterações em vídeos de entrevistas reais, sem

manipulação de IA.

A avaliação será feita em vídeos com duas qualidades de imagem distintas:

- 360p: Para verificar o desempenho do sistema em resoluções mais baixas.
- 1080p: Para verificar o desempenho do sistema em resoluções mais altas.

O teste busca determinar se a qualidade do vídeo interfere nos resultados da análise.

Podemos verificar através das tabelas 12, 13, 14, 15 ,16 ,17 e 18.

Tabela 12 - Teste 2 Redmi 9

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	144,049	107,229
Máximo:	193,473	214,669
Média:	~164,196	~126,515
Mediana:	163,58	114,19
Desvio-padrão:	~15,31	~32,01
Score		
Variação (mín e máx):	0,0323 e 0,4718	0,0184 e 0,3255
Média geral:	~0,1956	~0,1729
Acurácia		
Acurácia geral:	100%	100%

Fonte: Autores (2025)

Tabela 13 - Teste 2 Poco X5 Pro

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	41,92 s	54,019 s
Máximo:	119,058 s	172,949 s
Média:	~66,014 s	~78,949 s
Mediana:	~58,666 s	~114,19 s
Desvio-padrão:	~21,01 s	~32,69 s
Score		
Variação (mín e máx):	0,0091 e 0,5609	0,0114 e 0,5281
Média geral:	~0,1015	~0,1027
Acurácia		
Acurácia geral:	90%	90,9%

Fonte: Autores (2025)

Tabela 14 - Teste 2 Galaxy A51

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	129,45 s	156,01 s
Máximo:	191,73 s	229,04 s
Média:	~151,74 s	~204,27 s
Mediana:	~147,75 s	~206,41 s
Desvio-padrão:	~19,85 s	~20,27 s
Score		
Variação (mín e máx):	0,0298 e 0,4374	0,0246 e 0,3218
Média geral:	~0,1750	~0,1339
Acurácia		
Acurácia geral:	100%	100%

Fonte: Autores (2025)

Tabela 15 - Teste 2 Pc 1 (Ryzen 5 5500G/16 GB RAM)

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	35,35s	44,48s
Máximo:	67,30s	93,34s
Média:	~53,43s	~65,39s
Mediana:	52,98s	58,64s
Desvio-padrão:	~8,76s	~16,06s
Score		
Variação (mín e máx):	~0,331,95 e 1,00	0,2528 e 1,00
Média geral:	~0,5342	~0,6539
Acurácia		
Acurácia geral:	70%	57,14%

Fonte: Autores (2025)

Tabela 16 - Teste 2 Pc 2 (Ryzen 5 5600 / 16 GB RAM / RTX3060)

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	46,06s	51,94s
Máximo:	115,06s	66,13s
Média:	~82,84s	~59,29s
Mediana:	81,04s	59,36s
Desvio-padrão:	~17,27s	~4,88s
Score		
Variação (mín e máx):	00,00 e 1,00	0,2436 e 1,00
Média geral:	~0,4862	~0,5062
Acurácia		
Acurácia geral:	70%	70%

Fonte: Autores (2025)

Tabela 17 - Teste 2 Galaxy A14

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	29,00s	44,97s
Máximo:	103,37s	111,50s
Média:	~60,63s	~75,03s
Mediana:	~63,46s	~70,14s
Desvio-padrão:	~15,78s	~19,45s
Score		
Variação (mín e máx):	0,2057 – 0,9999	0,2456 – 0,9999
Média geral:	~0,5993	~0,5895
Acurácia		
Acurácia geral:	70%	60%

Fonte: Autores (2025)

Tabela 18 - Teste 2 Motorola G54

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	8,00s	85,83s
Máximo:	17,00s	263,45s
Média:	10,50s	82,15s
Mediana:	10,00s	90,53s
Desvio-padrão:	2,09s	40,00s
Score		
Variação (mín e máx):	0,0102 – 0,9968	0,0193 – 0,9968
Média geral:	~0,677	~0,677
Acurácia		
Acurácia geral:	90%	100%

Fonte: Autores (2025)

6.5 Teste 3 (Veo3)

Este teste utiliza vídeos gerados pela tecnologia **Veo3 do Google** para avaliar a capacidade do nosso projeto em identificar adulterações. O foco é analisar vídeos 100% sintéticos (falsos), com a inclusão de rostos e cenários criados por IA. Podemos verificar através das tabelas 19, 20, 21, 22, 23 e 24.

Tabela 19 - Teste 3 Redmi 9

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	155,388	116,006
Máximo:	516,195	246,231
Média:	~189,455	~140,593
Mediana:	169,451	123,052
Desvio-padrão:	~111,81	~42,12
Score		
Variação (mín e máx):	0,0075 e 0,9450	0,0070 e 0,9779
Média geral:	~0,3593	~0,2853
Acurácia		
Acurácia geral:	30%	10%

Fonte: Autores (2025)

Tabela 20- Teste 3 Poco X5 Pro

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	32, 656 s	48,865 s
Máximo:	70,832 s	79,133 s
Média:	55,702 s	65,718 s
Mediana:	55,852 s	66,446 s
Desvio-padrão:	~10,82 s	~13,94 s
Score		
Variação (mín e máx):	0,0451 e 0,6812	0,0434 e 0,7295
Média geral:	~0,3473	~0,3225
Acurácia		
Acurácia geral:	20%	30%

Fonte: Autores (2025)

Tabela 21 - Teste 3 Galaxy A51

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	129,41 s	180,67 s
Máximo:	413,47 s	471,13 s
Média:	~173,27 s	~228,31 s
Mediana:	149,96 s	207,19 s
Desvio-padrão:	~85,10 s	~85,96 s
Score		
Variação (mín e máx):	0,0073 e 0,9299	0,0066 e 0,9672
Média geral:	~0,3542	~0,23,03
Acurácia		
Acurácia geral:	30%	10%

Fonte: Autores (2025)

Tabela 22 - Teste 3 Pc 1 (Ryzen 5 5500G/16GB RAM)

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	22,49s	23,24s
Máximo:	63,15s	33,56s
Média:	~43,82s	~20,24s
Mediana:	45,67s	26,65s
Desvio-padrão:	~13,78s	~19,34
Score		
Variação (mín e máx):	0,2378 e 0,9732	0,3000 e 1,00
Média geral:	~0,4382	~0,3780
Acurácia		
Acurácia geral:	71,42%	87,53%

Fonte: Autores (2025)

Tabela 23- Teste 2 Pc 2 (Ryzen 5 5600 / 16 GB RAM / RTX3060)

	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	44,31s	46,31s
Máximo:	222,39s	108,24s
Média:	~78,64s	~54,76s
Mediana:	68,04s	48,70s
Desvio-padrão:	~53,17s	~18,91s
Score		
Variação (mín e máx):	0,3390 e 1,00	0,2500 e 1,00
Média geral:	~0,8712	~0,7274
Acurácia		
Acurácia geral:	80%	60%

Fonte: Autores (2025)

Tabela 24 - Teste 3 Galaxy A14

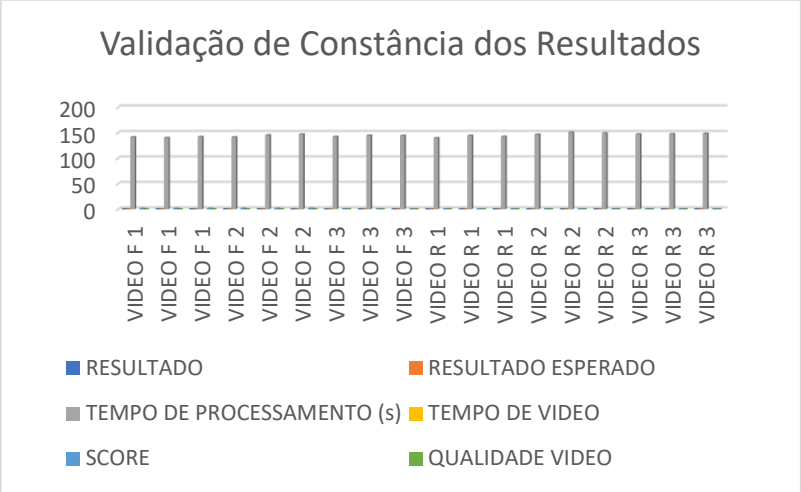
	Qualidade 360p	**Qualidade 1080p**
Tempo de Processamento (s)		
Mínimo:	64,958s	72,134s
Máximo:	128,462s	139,497s
Média:	~88,742s	~92,012s
Mediana:	85,671s	87,553s
Desvio-padrão:	~20,58s	~23,11s
Score		
Variação (mín e máx):	0,3032 – 0,8851	0,3445 – 0,9579
Média geral:	0,5321	~0,5653
Acurácia		
Acurácia geral:	68,3%	65,2%

Fonte: Autores (2025)

6.6 Teste Comparativo

O objetivo deste teste é verificar a confiabilidade e a consistência do nosso aplicativo. Para isso, analisaremos se ele mantém os resultados, mesmo ao processar o mesmo vídeo múltiplas vezes. Verificamos isso com a Tabela/Gráfico 25.

Tabela 25 - Vídeos compartilhados



Fonte: Autores (2025)

7. CONCLUSÃO

7.1 Teste – 1 CelebDF

Objetivo: avaliar desempenho em vídeos reais e falsos já rotulados.

- **Redmi 9**: desempenho excelente (100% de acurácia global), mas com tempo de processamento alto (~157s).
- **Poco X5 Pro**: processou mais rápido (~43s), mas a acurácia caiu para **85%**, com dificuldades em identificar alguns vídeos reais.
- **Galaxy A51 e PC1**: consistentes, em torno de **90% de acurácia**, porém com tempo mais elevado (~140s).
- **Galaxy A14**: o pior resultado, com apenas 55% de acurácia global, mostrando baixa confiabilidade no dispositivo.
- **Motorola G54**: apresentou 90% de acurácia, mas com scores muito baixos para fakes (~0,02), sugerindo dificuldade na calibragem da confiança.

Resumo Teste 1:

O desempenho varia bastante conforme o hardware. Dispositivos de entrada (ex: Galaxy A14) têm quedas severas na acurácia, enquanto intermediários e PCs mantêm resultados mais confiáveis.

7.2 Teste – 2 Impacto da qualidade (360p vs 1080p, cenário Real)

Objetivo: avaliar impacto da resolução de vídeo no desempenho.

- **Redmi 9**: manteve 100% de acurácia em ambas resoluções, mas com tempos bem diferentes (360p ~164s vs 1080p ~126s).
- **Poco X5 Pro**: resultados consistentes (≈90% de acurácia), mas com maior tempo em 1080p.
- **Galaxy A51**: manteve 100% de acurácia, porém o tempo aumentou bastante em 1080p (~204s).
- **PC1**: queda significativa de acurácia (de 70% em 360p para 57% em 1080p).
- **PC2 (com GPU RTX3060)**: estável em 70% de acurácia, independentemente da resolução, com processamento bem mais rápido.

- **Galaxy A14**: desempenho instável (70% em 360p, caiu para 60% em 1080p).
- **Motorola G54**: se destacou em 1080p (**100% de acurácia**), mas com maior variação no tempo de execução.

Resumo Teste 2:

A qualidade do vídeo influencia o desempenho de forma distinta em cada dispositivo. Smartphones médios e de entrada sofrem variações maiores, enquanto PCs com GPU e celulares mais equilibrados conseguem manter resultados consistentes.

7.3 Teste – 3 Impacto da qualidade (360p vs 1080p, Vídeos Veo3)

Objetivo: analisar capacidade do sistema em identificar vídeos totalmente artificiais.

- **Redmi 9 e Galaxy A51**: desempenho ruim, com acurácia caindo para **10–30%**.
- **Poco X5 Pro**: também fraco (**20–30% de acurácia**).
- **PC1 (Ryzen 5 5500G)**: desempenho sólido, com 71–87% de acurácia.
- **PC2 (Ryzen 5 5600 + RTX3060)**: ainda melhor, chegando a 80% em 360p e 60% em 1080p.
- **Galaxy A14**: apresentou 65–68% de acurácia, melhor que outros celulares, mas inferior ao PC com GPU.

Resumo Teste 3:

Esse foi o cenário mais desafiador. Dispositivos móveis tiveram dificuldades severas para identificar *deepfakes* sintéticos, enquanto PCs com GPU alcançaram resultados significativamente melhores, mostrando que a robustez do hardware impacta diretamente a eficácia contra *deepfakes* mais avançados.

Esse é o cenário que evidencia as limitações do modelo quando o conteúdo foge do padrão da base de treinamento, os erros aumentam drasticamente.

7.4 Teste – 4 Teste Comparativo (*Fake* vs Real)

- **Vídeos Fake**: média de score alta, como esperado, mas com **acurácia menor (70%)**, mostrando que ainda houve falsos negativos.
- **Vídeos Reais**: média de score muito baixa (~0,05) e acurácia perfeita (100%), com separação bem definida.

O modelo é mais robusto para detectar vídeos reais do que fakes, embora consiga identificar falsos na maioria dos casos.

Conclusão geral

Os testes mostram que:

- O modelo é muito eficiente em cenários controlados, com acurácia de até 100%.
- A acurácia cai bastante quando exposto a vídeos com ruído, variações ou complexidade maior, chegando a apenas 10% em 1080p.
- O tempo de processamento médio se manteve entre 140–190 s, mas pode ultrapassar 500 s em casos críticos, evidenciando a limitação de hardware.
- A separação entre scores de vídeos reais e falsos é clara na maior parte dos testes, mas existe uma zona cinzenta (scores intermediários) que gera erros de classificação.
- O modelo apresenta limitações claras quando o conteúdo foge do padrão da base de treinamento, os erros aumentam.

8. TRABALHOS FUTUROS

Abaixo, são indicadas algumas possíveis abordagens para pesquisas futuras, acompanhadas de explicações detalhadas.

8.1 Melhoria de desempenho:

A criação do aplicativo para reconhecer *deepfakes* apresentou resultados positivos, mas, devido ao caráter dinâmico e em permanente mudança das metodologias de inteligência artificial, é crucial sugerir aperfeiçoamentos constantes para assegurar sua pertinência e eficiência ao longo do tempo.

8.2 Monitoramento do progresso das IA:

Anualmente, surgem novas estruturas de redes neurais e recursos capazes de criar conteúdo mais realistas. Inovações como GANs melhoradas, modelos de difusão e plataformas multimodais têm aumentado a complexidade dos *deepfakes*, tornando a identificação um desafio em constante evolução. Diante disso, o aplicativo precisará receber atualizações regulares, integrando novas fontes de dados, técnicas de treinamento e as mais recentes arquiteturas de aprendizado profundo. Essa evolução incessante é crucial para assegurar a eficácia da detecção diante de manipulações cada vez mais sofisticadas e realistas.

8.3 Melhoria da aparência e da interação do usuário (UX/UI):

Uma interface bem elaborada é essencial para assegurar o envolvimento do usuário. Projetos futuros poderão concentrar-se na reformulação da interface do aplicativo, integrando elementos de design mais limpo, flexíveis e ajustáveis a diversas plataformas móveis. A implementação de práticas eficazes de UX (Experiência do Usuário) tornará a operação do sistema mais intuitiva, mesmo para aqueles com conhecimentos técnicos limitados.

8.4 Ampliação da detecção para imagens:

O foco inicial do sistema era a avaliação de vídeos, no entanto, as intervenções com inteligência artificial não se restringem a esse tipo de mídia. Os *deepfakes* em imagens têm aumentado em aceitação e podem ser igualmente prejudiciais, especialmente quando aplicados em plataformas sociais e desinformações. Projetos futuros devem expandir a funcionalidade do aplicativo para que os usuários enviem não apenas vídeos, mas também imagens estáticas. Essa modificação demandará ajustes no fluxo de pré-processamento e na estrutura do modelo, contudo, permitirá que o sistema se torne mais abrangente, identificando tanto vídeos quanto fotos alteradas.

8.5 Ampliação do escopo de análise para cenários:

Embora a identificação facial seja o foco principal do projeto atual, já existem tecnologias de inteligência artificial que podem criar cenários completos ou alterar componentes do ambiente de forma realista. Isso envolve, por exemplo, a criação de contextos falsos em vídeos e imagens, como cenários artificiais, objetos que não existem ou modificações nas expressões corporais. Dessa forma, uma possível melhoria do projeto é ampliar a análise para além das faces, possibilitando a detecção de alterações em outros aspectos visuais do conteúdo. Essa estratégia adicionaria força ao aplicativo, tornando-o ainda mais valioso em diversas situações de uso.

8.6 Integração com outras modalidades de análise (áudio e multimodalidade):

Outra abordagem seria a inclusão da avaliação de áudio e multimodalidade. Vários *deepfakes* misturam imagens forjadas com vozes geradas por inteligência artificial, resultando em efeitos extremamente credíveis. O sistema poderia ser melhorado para identificar discrepâncias na voz ou desajustes entre a fala e os movimentos dos lábios, aumentando a exatidão do diagnóstico geral. Essa combinação de modalidades significaria um progresso notável, aumentando a confiança na ferramenta.

8.7 Barra de progresso com porcentagem no carregamento de vídeo:

Implementar a exibição em tempo real do carregamento e processamento do vídeo na aplicação, para proporcionar transparência ao usuário e melhorar a sua experiência de uso. Dessa maneira o usuário compreenderia o que está acontecendo e teria mais confiança no sistema.

Além disso, essa funcionalidade permitiria ao time técnico monitorar o desempenho, identificar gargalos e otimizar o processamento.

Em resumo, implementar essa visibilidade transforma o *upload* em uma experiência interativa e confiável, fortalecendo tanto o engajamento do usuário quanto a eficiência operacional da aplicação.

8.8 Exemplos de uso

Para demonstrar a aplicabilidade prática do sistema desenvolvido, apresentam-se a seguir alguns exemplos de uso que simulam situações reais de utilização. Esses exemplos evidenciam como o sistema de detecção de *deepfake* pode ser empregado em diferentes contextos, contribuindo para a verificação da autenticidade de vídeos e para a redução da disseminação de conteúdos falsos.

- **Verificação de vídeos compartilhados em redes sociais**

Um usuário recebe um vídeo duvidoso pelo *WhatsApp*, supostamente mostrando uma figura pública fazendo uma declaração polêmica.

Ele faz o *upload* do vídeo no sistema e recebe o resultado indicando alta probabilidade de manipulação (95%).

Com base nisso, decide não repassar o conteúdo, ajudando a evitar a disseminação de fake *news*.

- **Jornalista checando autenticidade de material de reportagem**

Um jornalista está preparando uma matéria sobre campanhas políticas e quer confirmar se um vídeo viral é verdadeiro.

Ao enviar o vídeo para o sistema, o resultado aponta baixa probabilidade de *deepfake* (8%), confirmando que o material é legítimo.

Isso aumenta a confiabilidade da reportagem e evita a divulgação de

informações falsas.

- **Plataforma de notícias integrando o sistema**

Um portal de notícias integra a API do sistema ao seu fluxo de publicação. Antes de qualquer vídeo ser publicado, o sistema faz automaticamente a análise e sinaliza conteúdos suspeitos.

Essa integração garante credibilidade ao veículo de comunicação e reduz o risco de veiculação de *deepfakes*.

REFERENCIAS

AFCHAR, Darius; NOZICK, Vincent; YANG, Junlin; ROSS, Olivier. MesoNet: a Compact Facial Video Forgery Detection Network. In: 2018 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, 2018. p. 1–7. DOI: 10.1109/WIFS.2018.8630761.

AGÊNCIA SP. Confira o impacto da inteligência artificial no mercado de trabalho e na educação, segundo especialistas da USP, 2025. Disponível em: https://www.agenciasp.sp.gov.br/confira-o-impacto-da-inteligencia-artificial-no-mercado-de-trabalho-e-na-educacao-segundo-especialistas-da-usp/
. Acesso em: 14 out. 2025.

ANDROID. Conheça o Android Studio, 2025. Disponível em: https://developer.android.com/studio/intro?hl=pt-br
. Acesso em: 16 ago. 2025.

ANDRADE, Emmanuel. Matplotlib: dando vida aos dados em Python. DIO, 2023. Disponível em: https://www.dio.me/articles/matplotlib-dando-vida-aos-dados-em-python
. Acesso em: 16 ago. 2025.

ASIMOV ACADEMY. O que é OpenCV e para que serve?, 2024. Disponível em: https://hub.asimov.academy/blog/o-que-e-opencv-e-para-que-serve/
. Acesso em: 16 ago. 2025.

ATLASSIAN. O que é Git. Disponível em: https://www.atlassian.com/br/git/tutorials/what-is-git
. Acesso em: 16 ago. 2025.

BCA – LEE. Deepfake: entenda como funciona essa tecnologia e os impactos legais. Brock, Camargo Advogados, 2024. Disponível em: https://lbca.com.br/tag/deepfake/
. Acesso em: 16 ago. 2025.

CHESNEY, R.; CITRON, D. Deep Fakes: A Looming Challenge for Privacy, Democracy, and National Security. California Law Review, v. 107, n. 6, p. 1753–1820, 2019. DOI: https://doi.org/10.2139/ssrn.3213954

DOLHARE, Shubham et al. Deepfake Detection Challenge: Results and Lessons Learned. Facebook AI, 2020. Disponível em: https://ai.facebook.com/blog/deepfake-detection-challenge-results-an-open-initiative/
. Acesso em: 22 abr. 2025.

DOLHANSKY, Brian ; BITTON, Joanna ; PFLAUN, Ben ; LU, Jikuo ; HOWES, Russ ; WANG, Menglin ; CANTON FERRER, Cristian. The DeepFake Detection Challenge, 2025. Disponível em: https://ui.adsabs.harvard.edu/abs/2020arXiv200607397D/abstract
. Acesso em: 16 de agosto 2025.

E-SAFER. Deepfakes e manipulação de mídia: quais os perigos dessa tecnologia? Disponível em: https://e-safer.com.br/deepfakes-e-manipulacao-de-midia-quais-os-perigos-dessa-tecnologia/
. Acesso em: 1 maio 2025.

EDUCAO E MÍDIA. Deepfake e os perigos da manipulação, 2020. Disponível em: https://www.gazetadopovo.com.br/vozes/educacao-e-midia/deepfake-e-os-perigos-da-manipulacao/. . Acesso em: 1 maio 2025.

FFMPEG. About FFmpeg. Disponível em: https://ffmpeg.org/about.html
. Acesso em: 16 ago. 2025.

FORBES BRASIL. Alerta deepfake: entenda como empresa de Hong Kong caiu em golpe milionário, 2024. Disponível em: https://forbes.com.br/forbes-tech/2024/02/alerta-deepfake-entenda-como-empresa-de-hong-kong-caiu-em-golpe-milionario/
. Acesso em: 1 maio 2025.

GITHUB. About GitHub and Git. Disponível em: https://docs.github.com/pt/get-started/start-your-journey/about-github-and-git
. Acesso em: 16 ago. 2025.

GOOGLE. Kit ML, 2025. Disponível em: https://developers.google.com/ml-kit/guides?hl=pt-br
. Acesso em: 1 maio 2025.

GOOGLE DEVELOPERS. *Kotlin for Android Developers*. Mountain View: Google, 2017. Disponível em: https://developer.android.com/kotlin. Acesso em: 20 out. 2025.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. Cambridge: MIT Press, 2016. Disponível em: https://www.deeplearningbook.org

GOODFELLOW, I. et al. Generative Adversarial Nets. In: Advances in Neural Information Processing Systems (NeurIPS 2014). Montreal: Curran Associates, 2014. Disponível em: https://papers.nips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

HASH TAG TREINAMENTOS. Biblioteca NumPy: o que é, vantagens e como usar, 2025. Disponível em: https://www.hashtagtreinamentos.com/biblioteca-numpy-no-python
. Acesso em: 16 ago. 2025.

JAWABREH, Ahmad. Explore the most advanced deep learning algorithm for face detection, 2023. Disponível em: https://medium.com/the-modern-scientist/multi-task-cascaded-convolutional-neural-network-mtcnn-a31d88f501c8
. Acesso em: 10 out. 2025.

JETBRAINS. *Kotlin Programming Language.* Praga: JetBrains, 2011. Disponível em: https://kotlinlang.org. Acesso em: 20 out. 2025.

JETBRAINS. *Kotlin Multiplatform Overview.* Praga: JetBrains, 2023. Disponível em: https://kotlinlang.org/lp/multiplatform. Acesso em: 20 out. 2025.

KINGLER, Nico. EfficientNet: Otimizando a eficiência do aprendizado profundo, 2024. Disponível em: https://viso.ai/deep-learning/efficientnet/
. Acesso em: 10 out. 2025.

LBCA – Lee, Brock, Camargo Advogados. Deepfakes: ética e (ab)uso na era da inteligência artificial, 2024. Disponível em: https://lbca.com.br/deepfakes-etica-e-abuso-na-era-da-inteligencia-artificial/
. Acesso em: 1 maio 2025.

LI, Yuezun; YANG, Xin; SUN, Pu; QI, Honggang; LYU, Siwei. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. 2019. Disponível em: https://arxiv.org/abs/1909.12962
. Acesso em: 13 set. 2025.

LI, Yuezun et al. Celeb-DF: A New Dataset for DeepFake Forensics. arXiv preprint arXiv:1909.12962, 2019. Disponível em: https://arxiv.org/abs/1909.12962
. Acesso em: 22 abr. 2025.

LUTZ, M**.** *Learning Python.* 5. ed. Sebastopol: O'Reilly Media, 2013.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. Nature, v. 521, p. 436–444, 2015. DOI: https://doi.org/10.1038/nature14539

MTCNN. Introduction to MTCNN, s.d. Disponível em: https://mtcnn.readthedocs.io/en/latest/introduction/
. Acesso em: 13 set. 2025.

NAFZIGER, M.; JEMEROV, D. *Kotlin in Action.* Shelter Island: Manning Publications, 2017. Acesso em: 16 ago. 2025.

NORION. Como a Inteligência Artificial está transformando diferentes setores, s.d. Disponível em: https://www.norion.ind.br/como-a-inteligencia-artificial-esta-transformando-diferentes-setores#:~:text=Algoritmos%20de%20IA%20s%C3%A3o%20usados%20para%20otimizar%20a%20log%C3%ADstica%20e,forma%20mais%20r%C3%A1pida%20e%20eficaz. Acesso em: 1 maio 2025.

RÖSSLER, Andreas; COZZOLINO, Davide; VERDOLIVA, Luisa; RIESS, Christian; THIES, Justus; NIESSNER, Matthias. FaceForensics++: Learning to Detect Manipulated Facial Images. 2019. Disponível em: https://arxiv.org/abs/1901.08971
. Acesso em: 16 ago. 2025.

RÖSSLER, Andreas et al. FaceForensics++: Learning to Detect Manipulated Facial Images. arXiv preprint arXiv:1901.08971, 2019. Disponível em:

https://arxiv.org/abs/1901.08971
. Acesso em: 22 abr. 2025.

SUSNJARA, Stephanie; SMALEY, Ian. Docker. IBM. Disponível em: https://www.ibm.com/br-pt/think/topics/docker
. Acesso em: 16 ago. 2025.

VARGHESE, Reuben Suju. Celeb DF (v2), 2022. Disponível em: https://www.kaggle.com/datasets/reubensuju/celeb-df-v2
. Acesso em: 16 ago. 2025.

VINICIUS. Aplicações da Inteligência Artificial na saúde: veja como a IA está transformando a medicina. Portal Telemedicina, 2023. Disponível em: https://portaltelemedicina.com.br/inteligencia-artificial-na-saude
. Acesso em: 14 out. 2025.

VAN ROSSUM, G. Python Tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdã, 1991. Disponível em: https://www.python.org/doc/essays/blurb/

YUEZUN LI. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics, 2020. Disponível em: https://github.com/yuezunli/celeb-deepfakeforensics
. Acesso em: 16 ago. 2020.

Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, Siwei Lyu; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 3207-3216

APÊNDICE

APÊNDICE A - Casos de Uso

Nesta sessão, vamos detalhar a descrição de cada funcionalidade com base nos casos de uso levantados no tópico “5.3.2 Descrição dos Casos de Uso”, como mostra a tabela 21:

Tabela 26 - Casos de Uso

Caso de Uso	Participante	Objetivo	Pré-condição	Etapas	Resultado Esperado
Acessar o Sistema	Usuário	Entrar no sistema para utilizar as funcionalidades disponíveis	—	—	Acesso garantido ao sistema
Subir Vídeos para Verificação	Usuário	Enviar um arquivo de vídeo para ser analisado	Sessão ativa no sistema	1. Usuário seleciona um vídeo do dispositivo 2. Vídeo é enviado ao sistema 3. Sistema confirma o envio e inicia a análise	Vídeo em processamento
Observar os Resultados da Análise	Usuário	Visualizar o diagnóstico gerado pela IA	Análise concluída pelo sistema	1. Sistema disponibiliza o resultado 2. Usuário visualiza se o vídeo é real ou deepfake 3. Sistema exibe score de confiança e tempo	Resultado exibido de forma clara
Encerrar a Sessão de Uso	Usuário	Finalizar a sessão e sair do sistema	—	—	Sessão encerrada com sucesso

Fonte: Autores (2025)

APÊNDICE B – MIT License