UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

HELENA CRISTO MARTINS

COMPREENSÃO DA VULNERABILIDADE HUMANA NA IDENTIFICAÇÃO DE *DEEPFAKES*

VIÇOSA
2022

HELENA CRISTO MARTINS

COMPREENSÃO DA VULNERABILIDADE HUMANA NA IDENTIFICAÇÃO DE *DEEPFAKES*

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II – e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientadora: Prof.ª Dra. Kétia Soares Moreira

VIÇOSA
2022

HELENA CRISTO MARTINS

COMPREENSÃO DA VULNERABILIDADE HUMANA NA IDENTIFICAÇÃO DE DEEPFAKES

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 01 de abril de 2022.

COMISSÃO EXAMINADORA

Profª. Dra. Kétia Soares Moreira - Orientadora
Universidade Federal de Viçosa - UFV

Prof. Dr. André Gomes Torres - Membro
Universidade Federal de Viçosa -UFV

Profª. Dra. Mara Cristina da Silveira Coelho - Membro
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET - MG

Dedico esse trabalho e todo o meu esforço durante esses anos de graduação à minha avó, que é e sempre será a minha pessoa favorita no mundo.

Agradecimentos

Em primeiro lugar, a Deus, pela minha vida e por ter permitido que eu tivesse saúde e determinação para ultrapassar todos os obstáculos encontrados ao longo desses anos de graduação.

Aos meus pais, Rosana e Ildeu, que são meus grandes exemplos de profissionalismo, dedicação e esforço. Agradeço por todo apoio, carinho e amor incondicionais que sempre me proporcionaram ao longo de toda minha vida. Em especial, agradeço meu pai por me despertar a paixão pela matemática, que me fez estar aqui hoje.

À minha irmã, Milena, minha alma gêmea e o grande amor da minha vida, que nunca deixou de torcer por mim e sempre foi a minha maior incentivadora para tudo, independente da distância física.

À minha amiga de infância, parceira dessa e de todas as vidas possíveis, Samilla, por todo o apoio, amizade, companheirismo e carinho comigo durante todos esses anos.

Às minhas tias, Daia e Corro, que foram minha base familiar nesses anos de Universidade, não me deixando faltar nada e sempre me proporcionando muito amor e carinho.

Às amizades que a UFV me proporcionou. Em especial, à Laura, Mariana, Taynara, Fernanda, Ana Luiza, Gabriel Calais, Eduardo e Yan, que fizeram a diferença durante esses anos de graduação, me apoiando e sempre torcendo por mim e pela minha felicidade.

Por fim, agradeço à minha orientadora, Kétia Soares Moreira, que mergulhou de cabeça na ideia comigo, me ajudando com tudo o que precisei, com muita paciência, profissionalismo e competência, tornando toda essa experiência ainda mais gratificante.

Resumo

Deepfake é uma tecnologia que usa inteligência artificial para criar vídeos falsos. Essa técnica, que já gerou desde conteúdos pornográficos com celebridades até discursos fictícios de políticos influentes, cresceu tanto nos últimos anos, de maneira descontrolada, que hoje se faz necessário estudos acerca de como é o seu funcionamento, a fim de combater a disceminação de informações falsas.

Posto isso, esse trabalho estuda uma maneira de criação de *deepfakes* abordado na literatura, através da implementação de um código em Python; cria-se uma base de dados de vídeos adulterados para, posteriormente, realizar análises, tanto objetivas, com a utilização da métrica PSNR *(Peak Signal to Noise Ratio)*; como subjetivas, com uma pesquisa referenciada em cima da métrica MOS *(Mean Opinion Score)*. Tudo isso com o propósito de se entender melhor a vulnerabilidade humana no julgamento desse procedimento.

Palavras- chaves: *deepfake*, vídeos falsos, base de dados, MOS, PSNR.

Abstract

Deepfake is a technology that uses artificial intelligence to create fake videos. This technique, which has already generated from pornographic content with celebrities to fictitious speeches by influential politicians, has grown so much in recent years, in an uncontrolled way, that today it is necessary to study how it works in order to combat the dissemination of information. false.

That said, this work studies a way of creating deepfakes addressed in the literature, through the implementation of a code in Python; creates a database of adulterated videos to later carry out objective analyses, using the PSNR metric; and subjective, using the MOS metric. All this with the purpose of better understanding human vulnerability regarding the judgment of this procedure.

Keywords: deepfake, fake videos, database, MOS, PSNR.

Sumário

1 Introdução do Trabalho 11

1.1 Objetivo Geral 12

1.2 Objetivos Específicos 13

2 Referencial Teórico 14

2.1 Redes Neurais 14

2.2 Inteligência Artificial 16

2.3 Machine Learning 16

2.4 Deep Learning 17

2.5 Google Colaboratory 19

2.6 MOS - Mean Opinion Score 19

2.7 PSNR - Peak Signal To Noise Ratio 20

3 Metodologia 23

3.1 Demo do artigo “First Order Motion Model for Image Animation” 23

3.2 Estruturação e execução do Código 25

3.3 Análise subjetiva 35

3.4 Análise objetiva 36

4 Resultados 39

5 Conclusão 44

Referências Bibliográficas 46

Lista de Figuras

Figura 1 - Comparação entre os processos de ML e DL. Fonte: [23] 18

Figura 2 - Comparação de performance entre as abordagens. Fonte: [24] 19

Figura 3 - Visão geral da abordagem adotada. Fonte: [2] com adaptações feitas pelo próprio Autor. 24

Figura 4 - Foto recortada. Fonte: Próprio Autor 29

Figura 5 - Vídeo cortado. Fonte: Próprio Autor 30

Figura 6 - Foto e vídeo redimensionados e colocados no mesmo frame. Fonte: Próprio Autor 32

Figura 7 - Processo de criação da *deepfake*. Fonte: Próprio Autor 34

Figura 8 - Deepfake final. Fonte: Próprio Autor 35

Figura 9 - Perguntas do Formulário. Fonte: Próprio Autor 36

Figura 10 - Gráfico com as notas da veracidade dos vídeos. Fonte: Próprio Autor 41

Figura 11 - Gráfico com as notas da qualidade dos vídeos. Fonte: Próprio Autor 41

Lista de Tabelas

Tabela 1 - Escala de pontuação MOS 20

Tabela 2 - Relação PSNR com MOS 22

Tabela 3 - Resultados do formulário para o Grupo1 39

Tabela 4 - Resultados do formulário para o Grupo2 40

Tabela 5 - Resultado do PSNR 42

1 Introdução do Trabalho

Em 2018 vazaram imagens do ex presidente Obama cometendo injúrias durante um anúncio de serviço público, causando revolta na população, pois estes acreditaram na veracidade do conteúdo, mas essas pessoas não sabiam que se tratava de um vídeo falso [30].

Embora as falsificações analógicas e digitais não sejam novas, as *deepfakes* aproveitam técnicas poderosas de aprendizado de máquina (*Machine Learning*) e inteligência artificial para manipular ou gerar conteúdo visual e de áudio com alto potencial de credibilidade.

Trata-se de um procedimento de síntese de imagens ou sons humanos baseada em técnicas de inteligência artificial. O nome em si, *deepfakes*, surgiu de um usuário anônimo da plataforma Reddit (compilado de fóruns que se organizam por temas e reúnem pessoas com interesses em comum), que atendia pelo nome de *deepfakes* (junção de *deep learning* e *fakes*) e que compartilhou os primeiros deepfakes colocando celebridades desconhecidas em videoclipes adultos. Ao compartilhar o código de computador que produziu esse material alterado, levou a uma explosão de conteúdo falso por conta do interesse generalizado da comunidade *Reddit* nesse tema [1].

Quando surgiram, os vídeos criados com a tecnologia *deepfake* eram mais facilmente reconhecidos como falsos. Observando-os com mais atenção era possível ver falhas, mas isso não os impediu de enganar muitas pessoas. Atualmente, com o desenvolvimento de redes neurais profundas avançadas e a disponibilidade de grande quantidade de dados, tornaram as imagens e vídeos forjados quase indistinguíveis para seres humanos e até mesmo para algoritmos de computador sofisticados. Uma das maiores problemáticas acerca desse assunto, é que essa tecnologia, por mais que tenha várias aplicabilidades úteis, pode ser utilizada também para fins perigosos. Elas podem difamar a reputação de celebridades, políticos ou até mesmo de desconhecidos, e disseminar mentiras. Podendo ainda serem utilizadas como ferramenta política, como o exemplo dado anteriormente do ex presidente Obama, onde em casos como esse, colocam um discurso que a pessoa nunca pronunciou, mas o fazem de uma forma verossímil a ponto de prejudicar sua candidatura, manchar sua imagem, dentre outros exemplos. Ou seja, hoje as pessoas podem facilmente estar assistindo o líder de um país fazer um discurso

convincente do líder de outro país, ou vice-versa. Esse truque de *deepfake* é bem-sucedido por duas razões principais, credibilidade e acessibilidade.

Nesse cenário, então, surge a necessidade de estudar como são criadas as *deepfakes* para assim, aperfeiçoar a detecção das mesmas a fim de evitar a propagação de informações falsas. Existem diversas técnicas que permitem essa criação, neste trabalho o método adotado foi a animação de imagens segundo o modelo proposto no artigo "First order motion model for image animation" [2], onde, uma imagem específica é animada com base em um vídeo de apoio, utilizando uma rede neural já treinada.

Assim, a julgar pela importância da detecção de *deepfakes*, bem como o entendimento acerca de seu surgimento e funcionamento, este trabalho faz uma análise e releitura do código compartilhado, através Google Colaboratory, do artigo científico especificado em [2] para a criação de dez vídeos falsos, com o propósito de criar uma base de dados de *deepfakes* para, posteriormente, verificar, através de análises, sua qualidade; ou seja, o quão imperceptíveis são. Essas análises são divididas em dois focos, a análise subjetiva, através da utilização da métrica MOS (*Mean Opinion Score*) [9,11], e a objetiva, pela métrica PSNR (*Peak Signal To Noise Ratio*) [9,10].

O trabalho a seguir encontra-se dividido em cinco capítulos, incluindo o atual. No capítulo 2 é realizada uma revisão bibliográfica, de modo a apresentar alguns conceitos necessários para a compreensão acerca de tudo o que está sendo realizado nesse projeto. O capítulo 3 descreve a metodologia utilizada a fim de alcançar com êxito os objetivos propostos, isto é, criar uma base de dados de vídeos manipulados e, posteriormente, analisá-los a fim de tentar compreender a vulnerabilidade humana na identificação de *deepfakes*. Já no capítulo 4 é realizada uma discussão acerca dos resultados obtidos. E, por fim, o capítulo 5 apresenta as considerações finais deste trabalho.

1.1 Objetivo Geral

Criar uma base de dados de *deepfakes* por meio de estudos metodológicos, e assim, compreender a vulnerabilidade humana na identificação delas.

1.2 Objetivos Específicos

- Estudar os métodos conhecidos da literatura para criação *deepfakes*;
- Criar uma base de dados e metodologia para análise de *deepfakes*;
- Verificar a similaridade entre amostras reais e ludibriadoras de vídeos;
- Realizar um teste de avaliação subjetiva com MOS sobre as amostras criadas;
- Analisar objetivamente a qualidade dos vídeos produzidos através do PSNR.

2 Referencial Teórico

Para um melhor entendimento acerca do assunto e realização do trabalho é fundamental recorrer à literatura, com a finalidade de revisar, entender e definir alguns conceitos que serão empregados durante o seu desenvolvimento.

2.1 Redes Neurais

São sistemas computacionais compostos por nós interconectados que funcionam de maneira similar aos neurônios do cérebro humano. Através do uso de algoritmos, são capazes de reconhecer padrões e correlações em dados e, com o tempo, aprender e melhorar continuamente [13].

Uma rede neural simples é composta por uma camada de entrada, outra de saída e uma camada oculta entre elas. Essas camadas são conectadas por nós e essas conexões formam uma rede de nós interconectados. Um nó é modelado conforme o comportamento de um neurônio humano. Eles são ativados quando há estímulos ou entradas suficientes. Essa ativação se propaga pela rede, criando uma resposta ao estímulo. As conexões entre esses neurônios artificiais agem como sinapses, o que faz os sinais serem transmitidos um para o outro. Há sinalização entre camadas conforme eles trafegam da primeira até a última e são processados no decorrer do caminho [13,14].

Quando confrontados com um problema para resolver, por exemplo, os neurônios realizam cálculos matemáticos para decidir se há ou não informações suficientes a serem enviadas para o próximo neurônio. Em suma, eles leem todos os dados e decidem onde está o relacionamento mais forte. Nos exemplos de redes mais simples, as entradas de dados recebidas são somadas, caso essa soma seja maior que um valor limítrofe, o neurônio dispara e ativa os neurônios conectados a ele.

À medida que o número de camadas ocultas em uma rede neural aumenta, formam-se redes neurais profundas. As arquiteturas de *deep learning* levam as redes neurais simples a outro nível. Cientistas de dados, ao usarem essas camadas, podem construir suas próprias redes

de *deep learning* que possibilitam o *machine learning*, no qual o computador pode ser treinado para realizar tarefas humanas com precisão, como reconhecer fala ou identificar imagens. Outrossim, o computador pode aprender sozinho a reconhecer padrões em várias camadas de processamento [14].

Existem tipos diferentes de redes neurais profundas e cada uma delas possui vantagens e desvantagens, dependendo do uso. São elas:

- Redes neurais convolucionais (RNCs), trata-se de uma rede neural artificial que aplica uma operação de convolução nos dados de entrada. Uma camada convolucional aplica uma matriz de pesos chamadas de filtros ou *kernels* ao longo de uma imagem, e esses filtros extraem recursos, identificando curvas, linhas, formas e outros padrões [15].

- Redes neurais recorrente (RNRs), são projetadas para reconhecer padrões em sequências de dados. São redes com *loops*, permitindo que as informações persistam. Elas geram modelos dinâmicos capazes de captar as relações sequenciais dos exemplos anteriores e integrá-las no atual processo. Ou seja, é como se a rede neural adquirisse capacidade de memória a curto prazo com a persistência de informações no processamento sequencial. Por isso, essas redes são excepcionalmente úteis para processar dados sequenciais [16].

- Redes neurais *feedforward*, consistem no processamento paralelo e distribuído de informações, onde o conhecimento é adquirido durante o aprendizado. Este conhecimento está armazenado nos pesos sinápticos das interconexões de cada neurônio, assemelhando-se assim, ao comportamento do cérebro humano. Tais redes são estáticas, pois suas saídas dependem em algum momento apenas das entradas [17,18].

- Redes neurais autoencoder são compostas por uma camada de entrada, uma de saída e uma intermediária, treinadas de forma não supervisionada para tentar copiar suas entradas para suas saídas. Uma estrutura de autoencoder inclui geralmente os processos de codificação e decodificação. Embora semelhante às redes neurais mais tradicionais, os autoencoders procuram modelar a entrada por si mesmos e, como tal, o método é considerado não supervisionado. Sua premissa é reduzir a sensibilidade ao conteúdo irrelevante e aumentá-la em relação ao conteúdo relevante. À medida que as camadas

são adicionadas, outras abstrações são formuladas como camadas superiores (aquelas mais próximas do ponto em que as camadas de decodificação são introduzidas). Essas abstrações podem então ser usadas por classificadores lineares ou não lineares [19].

2.2 Inteligência Artificial

Inteligência artificial (IA) é um termo amplo que abrange qualquer tipo de implementação computacional que tente imitar características cognitivas humanas, como, por exemplo, resolução de problemas. Isto é, trata-se de um ramo da ciência da computação que se propõe a desenvolver sistemas que simulem a capacidade humana na percepção de um problema, identificando seus componentes e, com isso, resolver problemas e propor ou tomar decisões [20,21].

IA envolve várias etapas ou competências como reconhecer padrões e imagens, entender linguagem escrita e falada, perceber relações e nexos, seguir algoritmos de decisão propostos, dentre outros exemplos. Ou seja, não apenas processar dados, mas sim, ser capaz de entender conceitos, adquirir raciocínios pela capacidade de integrar novas experiências e, assim, se auto aperfeiçoar, resolvendo problemas, ou até mesmo realizando tarefas [20]. De forma prática, a área de IA está organizada em três eixos: representação do conhecimento, tomada de decisão e aprendizado. A representação do conhecimento é o domínio da epistemologia, da mesma forma, a tomada de decisão consiste no processo decisório da escolha dentre alternativas possíveis, ou seja, opções pré-existentes. Já o *Machine Learning* lida com técnicas estatísticas de processamento de dados. Sistemas computadorizados de apoio à decisão existem há décadas, mas o aumento da velocidade de processamento e de armazenamento de informação dos computadores, permitiu analisar um grande volume de dados em um tempo muito mais curto, propondo soluções de problemas, orientando a proposta e tomada de decisões, realizando tarefas sem interferência humana [5,21].

2.3 Machine Learning

Traduzido do inglês como Aprendizado de Máquina, trata-se de um ramo da inteligência artificial baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana possível e isso é feito a partir do

treinamento de um modelo matemático que é alimentado por dados estruturados. Ou seja, trata-se de uma técnica onde uma máquina adquire conhecimento com base em aprendizado [22].

Técnicas baseadas em aprendizado de máquina têm sido empregadas com sucesso em diversos campos como, por exemplo, aplicações biomédicas e médicas. Mais da metade dos pacientes com câncer recebe radiação ionizante (radioterapia) como parte de seu tratamento, sendo a principal modalidade de tratamento em estágios avançados da doença local. A radioterapia envolve um grande conjunto de processos que não apenas abrangem o período da consulta ao tratamento, mas também se estendem além disso para garantir que os pacientes tenham recebido a dose de radiação prescrita e estejam respondendo bem. Os graus de complexidade desses processos podem variar e podem envolver vários estágios de interações homem-máquina e tomada de decisões, o que naturalmente sugere o uso de algoritmos de aprendizado de máquina para otimizar e automatizar esses processos, incluindo, entre outros, garantia de qualidade da física de radiação, contorno e planejamento de tratamento, radioterapia guiada por imagem, gerenciamento de movimento respiratório, modelagem de resposta ao tratamento e previsão de resultados [6,22].

2.4 Deep Learning

É uma técnica específica dentro da área de *Machine Learning* que também envolve a construção de algoritmos que usam dados para aprender a resolver determinadas tarefas. Uma das principais diferenças é que o *Deep Learning* é intuitivo, enquanto o *Machine Learning* exige uma intervenção manual na seleção dos recursos a serem processados. A principal ideia em *Deep learning* é imitar a rede de neurônios presentes nos humanos, então sua implementação é feita com Redes Neurais Artificiais [22].

No caso das redes neurais humanas, para realizar determinada tarefa, um neurônio recebe um impulso elétrico e então ativa n outros neurônios conectados a ele, e assim sucessivamente, até que determinada tarefa seja realizada. No caso da ANN (*Artificial Neural Network*), cada "ponto" da rede é chamado de neurônio e, dependendo dos dados de entrada, esses neurônios serão ativados e, então, ativarão os neurônios subsequentes na rede neural, de acordo com os dados fornecidos [12]. Logo, cada camada contém unidades que transformam os dados de entrada em informações que a próxima camada pode usar em sua tarefa de predição. Com essa estrutura, as máquinas podem aprender durante seu próprio processamento de dados.

Ou seja, à medida que o dado de entrada passa por cada uma das diversas camadas da rede neural, a estrutura da rede vai definir atributos específicos daqueles dados, de forma hierárquica. Então, o que acontece é que a etapa de extração de atributos é realizada pela própria rede neural, além dela também realizar a classificação, diferentemente do que ocorre no *Machine Learning*, onde essa etapa é realizada com intervenção humana [23]. Essa diferença está representada na Figura 1 abaixo.

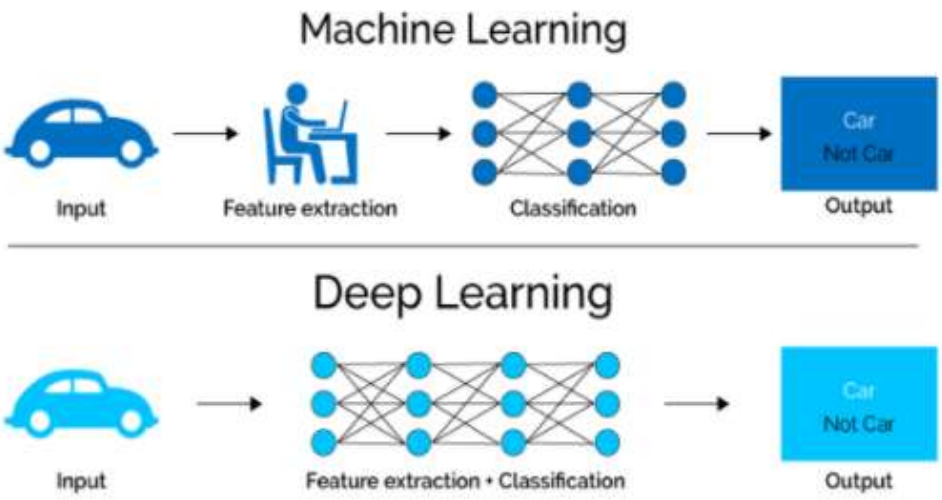

Figura 1 - Comparação entre os processos de ML e DL. Fonte: [23]

Logo, utilizar *Deep Learning* vai ser uma boa escolha quando tiver uma quantidade muito grande de dados e operações mais complexas já que devido à sua estrutura de camadas e muitas interações entre os neurônios, ele exige um poder computacional muito maior. Já o *Machine Learning* vai ser uma boa alternativa quando tiver uma quantidade menor de dados e o custo operacional de transformá-los em dados estruturados não for grande [24]. A Figura 2 abaixo mostra que a aplicação de redes neurais artificiais é interessante para larga escala, para casos em que escalabilidade é importante e casos em que se tem muitos dados. Já o *Machine Learning* tradicional, acaba sendo a melhor opção com uma quantidade menor de dados [24].

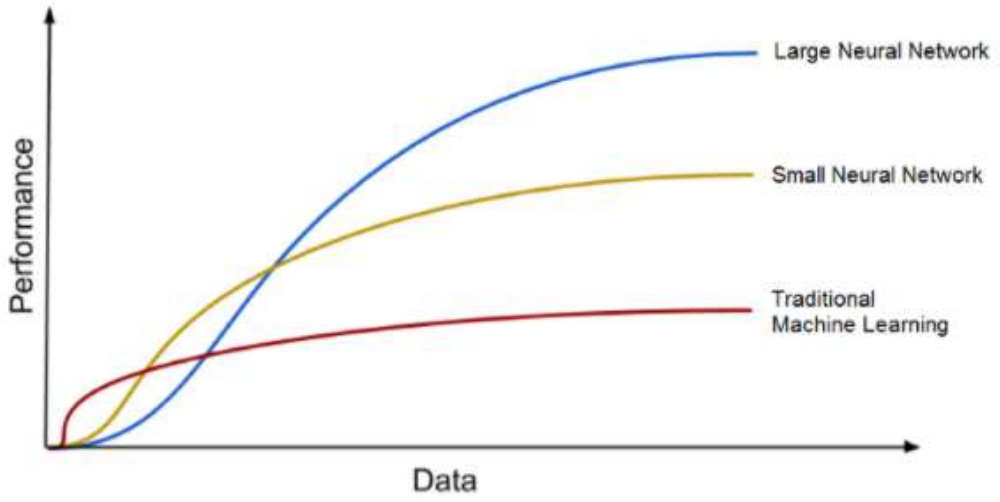

Figura 2 - Comparação de performance entre as abordagens. Fonte: [24]

2.5 Google Colaboratory

Também conhecido como Colab, é um serviço de nuvem gratuito hospedado pelo próprio Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial.

Trata-se de uma ferramenta que permite que o usuário misture código fonte, geralmente em *python*, e texto rico, geralmente em *markdown*, com imagens e o resultado desse código. É uma técnica conhecida como *notebook* (“caderno”). Tudo isso em um ambiente colaborativo, onde o usuário pode compartilhar com outras pessoas, permitindo que outros rodem seu código e até o modifiquem criando suas próprias versões. Essa ferramenta trabalha em especial com a linguagem *Python*, mas com alguns ajustes é possível rodar código em R, Julia, *Swift* e outras linguagens [7].

2.6 MOS - Mean Opinion Score

O MOS é uma métrica subjetiva que consiste na obtenção de um valor para determinar a qualidade de um vídeo através das opiniões dos usuários. Essa pontuação pode ser empregada em qualquer situação onde a experiência e a opinião subjetiva humana sejam úteis [9]. Na prática, é frequentemente usado para julgar aproximações digitais de fenômenos mundiais.

Essa métrica tem como base uma tabela pré determinada, como mostra a Tabela 1 a seguir, que é baseada na exibição de vídeos onde os usuários possam avaliar e pontuar em uma escala de 1 a 5 a qualidade desse vídeo.

Tabela 1 - Escala de pontuação MOS

MOS	AVALIAÇÃO
5	Excelente
4	Bom
3	Razoável
2	Pobre
1	Ruim

Fonte: Elaborado pelo Autor

Devido à tendência humana de evitar classificações perfeitas, uma média em torno de 4,3 e 4,5 é considerado um alvo de excelente qualidade. Na extremidade inferior, a qualidade da chamada ou do vídeo torna-se inaceitável abaixo de um MOS de aproximadamente 3,5.

Por mais que se trate de uma métrica subjetiva, MOS também foi relacionada com a PSNR, explicado na próxima seção, para que fosse possível obter uma avaliação do MOS a partir de uma avaliação objetiva, sem os processos trabalhosos que dependem de opiniões humanas, por exemplo [9].

2.7 PSNR - Peak Signal To Noise Ratio

A métrica PSNR é uma derivação da métrica *Mean Square Error* (MSE), que faz a relação de um quadro original, isto é uma imagem, com o quadro recebido [9]. Ou seja, relaciona a potência máxima do sinal com a potência do ruído, tentando comparar o sinal antes e depois de algum processo de degradação e então, gera o resultado em dB (decibel). No procedimento de transmissão de vídeo, a PSNR relaciona a entrada e saída de compressão de perdas que avalia se houve ou não ruído introduzida na imagem ou frames.

Como mostra Gracieth Mendes Valenzuela em [9], matematicamente o MSE e PSNR são definidos como:

Equação (1): $$MSE = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left\| Ys_{(i,j)} - Yd_{(i,j)} \right\|^2$$

Em que:

MSE – Mean Square Error;

$Ys_{(i,j)}$ – Posição do pixel no quadro original;

$Yd_{(i,j)}$ – Posição do pixel no quadro degradada;

M x N – Quantidade de pixels do quadro.

Equação (2): $$PSNR = 20log_{10}\left[\frac{255}{\sqrt{\frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left\| Ys_{(i,j)} - Yd_{(i,j)} \right\|^2}}\right]$$

O PSNR é uma derivação do MSE em relação ao valor máximo de luminosidade para cada pixel. Considerando 8 bits por pixel, obtém-se 255 ($2^8 - 1$) valores de luminosidade.

Para que um vídeo seja considerado com qualidade satisfatória é necessário que o valor médio do PSNR esteja entre 25 e 31 dB, como ilustrado na Tabela 2 que apresenta relação entre os valores de PSNR para MOS. Valores acima de 37dB (decibéis) correspondem a perdas imperceptíveis ao olho humano, com uma qualidade excelente. PSNR com valor de 31 dB a 37dB apresenta qualidade boa, de 20dB a 25dB com qualidade pobre e abaixo de 20dB a qualidade é ruim [9].

Tabela 2 - Relação PSNR com MOS

PSNR	MOS	QUALIDADE
Maior que 37dB	5	Excelente
Entre 31 e 37 dB	4	Bom
Entre 25 e 31 dB	3	Razoável
Entre 20 e 25 dB	2	Pobre
Menor que 20 dB	1	Ruim

Fonte: [27]

3 Metodologia

A seguir encontram-se detalhadas as etapas executadas para a criação da base de dados de vídeos alterados e as análises, subjetiva e objetiva, dos mesmos.

3.1 Demo do artigo "First Order Motion Model for Image Animation"

O método utilizado para criação das *deepfakes* desse trabalho foi proposto no artigo cientifico "*First Order Motion Model for Image Animation*" [2], que segue uma estratégia auto-supervisionada inspirada na rede neural Monkey-Net [27].

A Figura 3 ilustra um resumo do procedimento abordado em [2]. O momento 1 mostra a imagem fonte S, *source*, que seria a imagem original, e um quadro de vídeo $\mathcal{D}$, *driving frame*, o vídeo base, como entradas. Já no momento 2, tem -se S e $\mathcal{D}$ passando por um detector de ponto-chave não supervisionado, ou seja, sem intervenção humana, o qual extrai a representação de movimento de primeira ordem consistindo em pontos-chave dispersos e transformações afins locais em relação a um quadro de referência $\mathcal{R}$. Em relação a essas tranformações afins locais, trata-se de uma ferramenta algébrica usada para correção/ajuste de imagem, que facilitam o emparelhamento de duas visões, melhoram significativamente a qualidade da estimativa e permitem modelar uma família maior de transformações [26]. Proseeguindo, no momento 3 a rede de movimento densa usa a representação de movimento para gerar fluxo óptico denso $\widehat{\mathcal{T}_{S\leftarrow\mathcal{D}}}$ de $\mathcal{D}$ para S e mapa de oclusão $\widehat{\mathcal{O}_{S\leftarrow\mathcal{D}}}$. Note que a rede de movimento densa nada mais é que uma rede *feedfoward* sendo usada para analisar/ trabalhar o movimento a ser empregado em S de $\mathcal{D}$. Por fim, no momento 4 a imagem de origem e as saídas da rede de movimento densa são usadas pelo gerador para renderizar a imagem de destino. O termo "renderizar" vem sendo usado na computação gráfica, significando converter uma série de símbolos gráficos em um arquivo visual, ou seja, "fixar" as imagens em um video, convertendo-as de um tipo de arquivo para outro, ou ainda "traduzir"de uma linguagem para outra.

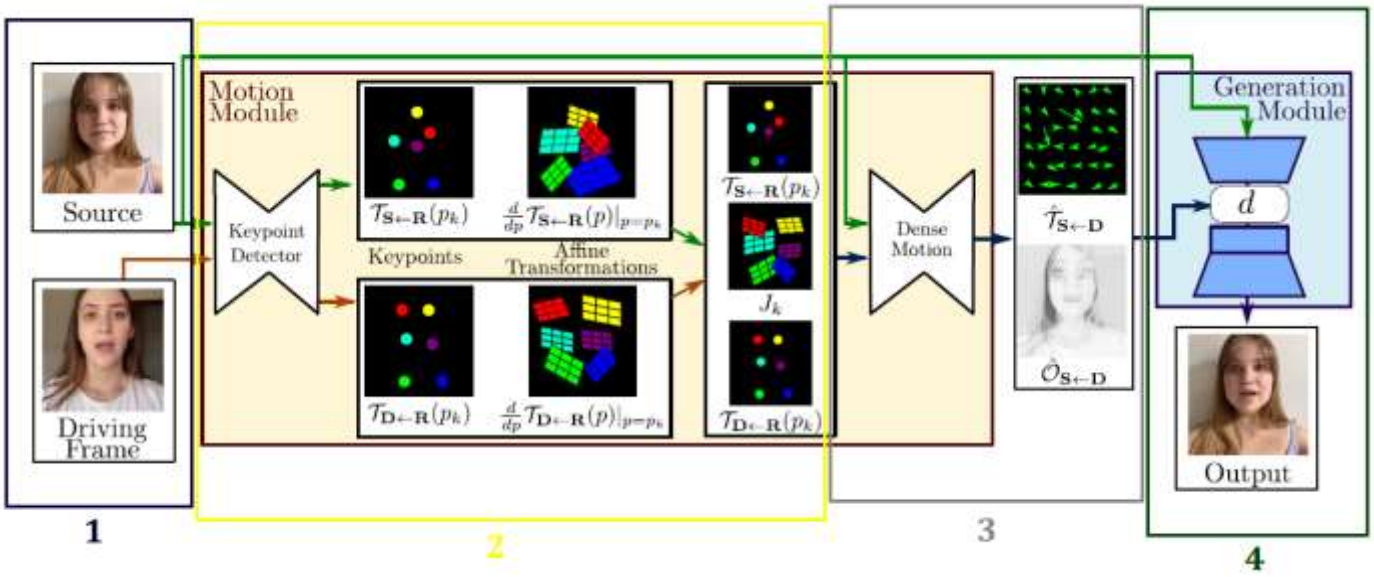

Figura 3 - Visão geral da abordagem adotada. Fonte: [2] com adaptações feitas pelo próprio Autor.

O *framework* é composto por dois módulos principais: o módulo de estimativa de movimento, *motion module*, representado no momento 2 da Figura 3; e o módulo de geração de imagens, *generation module*, representado no momento 4. O objetivo do primeiro é prever um campo de movimento denso a partir do $\mathcal{D}$ pertencente a $\mathbb{R}^{3\times H\times W}$ da dimensão H x W de $\mathcal{D}$ para S pertencente a $\mathbb{R}^{3\times H\times W}$. O campo de movimento denso é usado posteriormente para alinhar os mapas de recursos calculados de S com a pose do objeto em $\mathcal{D}$. O campo de movimento é modelado por uma função $\mathcal{T}_{S\leftarrow\mathcal{D}}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ que mapeia cada localização de pixel em $\mathcal{D}$ com sua localização correspondente em S. $\mathcal{T}_{S\leftarrow\mathcal{D}}$ é muitas vezes referido como fluxo óptico inverso. É empregado fluxo óptico inverso, em vez de fluxo óptico direto, pois a deformação pode ser implementada eficientemente de maneira diferenciável usando amostragem bilinear [8]. Salienta-se que fluxo óptico é a distribuição das componentes de velocidade dos padrões de brilho em uma imagem, isto é, um campo de vetores de velocidade associado a uma sequência de imagem. Nesse sentido, o fluxo óptico consiste em um campo denso de velocidade em que cada pixel no plano da imagem está associado a um único vetor velocidade [28]. Dando continuidade, assume-se que existe um referencial abstrato $\mathcal{R}$. Estima-se independentemente duas transformações: de $\mathcal{R}$ para S ($\mathcal{T}_{S\leftarrow\mathcal{R}}$) e de $\mathcal{R}$ a $\mathcal{D}$ ($\mathcal{T}_{\mathcal{D}\leftarrow\mathcal{R}}$). O referencial é um conceito abstrato que se cancela nas derivações posteriores. Portanto, nunca é calculado explicitamente e não pode ser visualizado. Essa escolha permite processar $\mathcal{D}$ e S de forma independente. Isso é desejado, pois, no momento do teste, o modelo recebe pares da imagem de origem e quadros de condução amostrados de um vídeo diferente, que pode ser

muito diferente visualmente. Em vez de prever diretamente $\mathcal{T}_{\mathcal{D}\leftarrow\mathcal{R}}$ e $\mathcal{T}_{S\leftarrow\mathcal{R}}$ o módulo estimador de movimento prossegue em duas etapas.

Na primeira etapa, ambas as transformações são aproximadas a partir de conjuntos de trajetórias dispersas, obtidas por meio de pontos-chave aprendidos de forma autossupervisionada. As localizações dos pontos-chave em $\mathcal{D}$ e S são previstas separadamente por uma rede codificador-decodificador. A representação do ponto-chave atua como um gargalo, resultando em uma representação de movimento compacta. Essa representação de movimento é adequada para animação, pois no momento do teste, os pontos-chave da imagem de origem podem ser movidos usando as trajetórias dos pontos-chave no vídeo de condução. Assim, modela-se o movimento na vizinhança de cada ponto-chave usando transformações afins locais. Em comparação com o uso de deslocamentos de ponto-chave apenas, as transformações afins locais permitem modelar uma família maior de transformações. É usada a expansão de Taylor para representar $\mathcal{T}_{\mathcal{D}\leftarrow\mathcal{R}}$ por um conjunto de localizações de pontos-chave e transformações afins. Para este fim, a rede de detectores de pontos-chave emite as localizações dos pontos-chave, bem como os parâmetros de cada transformação afim.

Durante a segunda etapa, uma rede de movimento denso combina as aproximações locais para obter o campo de movimento denso resultante $\widehat{\mathcal{T}_{S\leftarrow\mathcal{D}}}$. Além disso, esta rede emite uma máscara de oclusão $\widehat{\mathcal{O}_{S\leftarrow\mathcal{D}}}$ que indica quais partes da imagem de $\mathcal{D}$ podem ser reconstruídas por distorção da imagem de origem e quais partes devem ser pintadas, ou seja, inferidas do contexto.

Finalmente, o módulo de geração renderiza uma imagem do objeto de origem em movimento conforme fornecido no vídeo de condução. Aqui, é usado uma rede geradora G que distorce a imagem de origem de acordo com $\widehat{\mathcal{T}_{S\leftarrow\mathcal{D}}}$ e pinta as partes da imagem que estão ocluídas na imagem de origem.

3.2 Estruturação e execução do Código

Para a criação das *deepfakes*, foi utilizado o código criado em [2], porém, automatizado e disponibilizado para uso com fins educacionais pelo programador e cofundador da Alura (plataforma brasileira de cursos de tecnologia), Guilherme Silveira [29].

Primeiramente é necessário baixar a rede neural, disponibilizada em [29] e já treinada como explicado em [2].

Após isso, organiza-se tudo em uma pasta no Google Drive, pois o Google Colab irá realizar uma comunicação com a mesma, ou seja, nessa pasta será colocada o arquivo de modelo da rede neural, a imagem que se deseja animar e o vídeo de apoio. Dessa forma, basta permitir que o Google Colab acesse o drive através do seguinte código:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Em seguida define-se o nome do arquivo e a extensão da imagem, que serão chamados posteriormente com o intuito de otimizar o processo, pois já deixa essa chamada no início do código.

```
base_name = "Irmã"
image_extension = "png"
```

Logo após, são definidas as dimensões do corte para garantir que haja centralização.

```
# corrija o corte do vídeo de origem. rode uma vez assim, depois vá atualizando

# assumindo que o vídeo é 1920 x 1080
# o rosto da pessoa está na parte "de baixo" do vídeo

DIMENSION = 1080
MIN_X = 0
MIN_Y = 640

should_crop = True
```

Depois, é realizado um setup de diretórios através do *"git clone"*, que tem como principal proposta clonar todos os arquivos de repositório remoto para um repositório local.

```
!git clone https://github.com/AliaksandrSiarohin/first-order-model
```

```
cd first-order-model
```

Dessa forma passa-se a ter todos os arquivos necessários do código original no repositório local. Note que o comando *"cd"* apenas exibe o nome do diretório.

Prontamente, são chamados a imagem e o vídeo, que foram previamente disponibilizados no drive, através dos comandos:

```
movie_input = base_name
movie_original = f"/content/gdrive/My Drive/deepfake/{movie_input}.mp4"
```

```
input = base_name
image_output = f"/content/gdrive/My Drive/deepfake/{input}-256.png"
```

Avisos em Python são gerados quando alguma classe, função ou palavra-chave desatualizada são usados, não são erros propriamente ditos. Logo, utiliza-se a função "*filterwarnings()"* para suprimir os avisos.

```
import warnings


warnings.filterwarnings("ignore")
```

Em seguida, são conferidas as dimensões da imagem. Primeiramente essa imagem é aberta utilizando a biblioteca de manipulação de imagens para Python chamada PIL. Posteriormente, é realizada uma verificação da altura (*height*) e da largura (*width*) da imagem.

```
from PIL import Image


im = Image.open(f"/content/gdrive/My Drive/deepfake/{input}.{image_extension}")
width, height = im.size
```

Se a altura e a largura forem diferentes (*"!="*), pega-se o menor dos dois, através do comando *"min(width, height)",* e, com o objetivo de tornar a imagem mais quadrada e centralizada, realiza-se algumas operações para cortá-la.

```
if width != height:
  new_size = min(width, height)
  left = (width - new_size)/2
  top = (height - new_size)/2
  right = (width + new_size)/2
  bottom = (height + new_size)/2
```

Assim, é definido o novo corte *("cut")* da imagem e, logo em seguida, executado através do comando *"crop"*.

```
cut = (left, top, right, bottom)
print(cut)
im = im.crop(cut)
```

No código original disponibilizado no artigo [2], foram utilizadas imagens de 256 por 256 pixels. Logo, como essa parte será reaproveitada, isso é necessário no atual código também, por isso, após certificar-se que a imagem está quadrada, é feito um processo para alterar as dimensões para 256 por 256 através do comando *"thumbnail"*. Em suma, esse comando é usado para fazer a imagem em miniatura da imagem dada. Após isso, essa imagem com dimensões alteradas de maneira correta para o código é salva e também mostrada como ficou através do display.

```
print(im.size)
im.thumbnail((256, 256))
print(im.size)
im.save(image_output)
display(im)
```

Note que, no caso de imagens com largura menor que 256 o código não irá rodar corretamente, pois não está definido como aumentar uma imagem e sim, apenas como cortar para diminuí-la. Logo, é recomendado utilizar uma imagem maior.

```
(0.0, 200.0, 1200.0, 1400.0)
Oh, a imagem que você pegou não é quadrada, cortando o meio... 1200 x 1600
(1200, 1200)
(256, 256)
```

Figura 4 - Foto recortada. Fonte: Próprio Autor

Já para o caso do vídeo, primeiro é verificado se a pessoa realmente deseja cortar, pois muitas vezes isso pode ser feito de maneira manual. Caso se deseje cortar automático, isso será feito através do comando "*if should_crop"*.

```
!apt install ffmpeg
import os
import shutil
from moviepy.editor import VideoFileClip

movie_source = "../movie_quadrado.mp4"
```

```
if should_crop:
  !ffmpeg -y -hide_banner -loglevel warning -i "$movie_original" -filter:v 'crop={DIMENSION}:{DIMENSION}:{MIN_X}:{MIN_Y}' "$movie_source"
else:
  shutil.copyfile(movie_original, movie_source)
```

Nele, será rodado um "*ffmpeg"*, que é um programa bem completo de recursos para conversão de vídeo e que normalmente já vem instalado nas máquinas virtuais do Google Colab. Mas, caso dê erro, basta instalar através do comando "*!apt install"*, onde *"apt"* é uma ferramenta que ajuda na instalação, atualização e desinstalação de programas. Seguido de *"install"*, é indicado ao *"apt"* que se deseja instalar o *"ffmpeg"*. Esse, por sua vez, é chamado através da utilização da exclamação, em seguida tem-se um *"-y"* para que não seja perguntado se gostaria que sobrescrevesse o arquivo se executado duas vezes, *"-hide_banner"* para tornar o processo silencioso, *"-loglevel warning"* para tornar ainda mais silencioso, isto é, menos

poluído no display, *"-i "* de *input* onde coloca-se o vídeo, em seguida *"-filter:v"* que trata-se de um filtro onde será realizado um corte nas dimensões e, por fim, o arquivo onde será colocado, que é a saída. Caso não se tenha interesse em realizar esse corte, o arquivo de origem é copiado para o arquivo final, como indicado no bloco *"else"*, através do comando *"copy"* e do módulo *"shutil"*, que se trata de um módulo que oferece várias operações de alto nível em arquivos e coleções de arquivos. Em particular, funções que possuem suporte a cópia e remoção de arquivos e isso só é possível através do import dessa biblioteca específica. Outra biblioteca também muito importante é a *"os",* que se trata de uma biblioteca de comandos do sistema operacional que auxilia a fazer algumas operações dentro do computador.

Através da biblioteca *"moviepy"* o vídeo que será utilizado para a criação da *deepfake*, já dentro dos parâmetros, é mostrado no display, através do comando *"ipython_display".*

```
from moviepy.editor import ipython_display

ipython_display(movie_source, width=256)
```

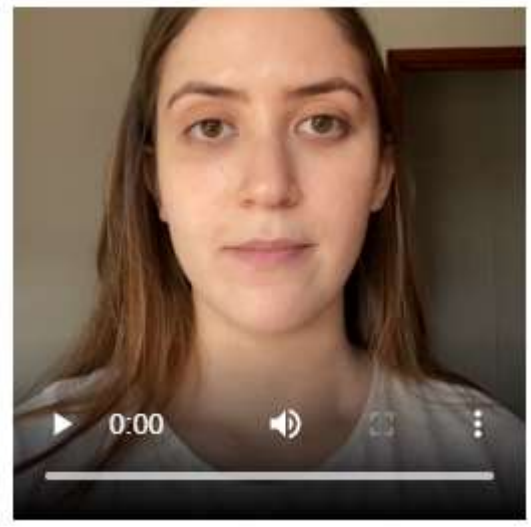

Figura 5 - Vídeo cortado. Fonte: Próprio Autor

Logo após, tem-se o processo de extração de um arquivo mp3 do vídeo, que é feito através, também, do "*ffmpeg"*.

```
generated_audio = f"../generated_audio_{base_name}.mp3"
!ffmpeg -y -hide_banner -loglevel warning -i "$movie_original" -vn -ar 44100 -ac 2 -ab 192k -f mp3 "$generated_audio"
```

A diferença nesse ponto, é que ao utilizar *"-vn"* indica-se que não se deseja um vídeo, com o uso de *"-ar 44100 -ac 2 -ab 192k"* especifica-se o interesse em características de áudio, onde 44100 é uma frequência de amostragem comum, 2 representa o número de canais e não é necessário mais do que um MP3 de 192 kbps para exportar áudio de ótima qualidade.

O próximo passo consiste em fazer a leitura da imagem juntamente com o vídeo, através do *"imread"* e *"mimread",* para que caso seja necessário, haja um redimensionamento, ou seja, trata-se de uma etapa de testes para verificar se está centralizado.

```
import imageio
from skimage.transform import resize
from tqdm.notebook import trange, tqdm

source_image = imageio.imread(image_output)
driving_video = imageio.mimread(movie_source, memtest=False)

source_image = resize(source_image, (256, 256))[..., :3]
driving_video = [resize(frame, (256, 256))[..., :3] for frame in tqdm(driving_video, desc="video resize")]
```

Então, pega-se a imagem trabalhada, faz um redimensionamento através do uso do comando *"resize"* para 256 por 256 pixels, em 3 canais de cores. A mesma coisa para o vídeo, onde para cada frame do mesmo foi feito um *"resize"*. Note que são realizados alguns imports, que tornam possível a utilização de determinados artifícios e/ou bibliotecas, como o *"imageio"* que é uma biblioteca Python que fornece uma interface fácil de ler e escrever uma ampla gama de dados de imagem, o *"resize"* explicado acima e o *"tqdm"* que se trata de uma biblioteca usada para criar barras de progresso.

Diferentemente do programa original, aqui é utilizado um *"memtest = False",* pois pode-se utilizar vídeos muito grandes, tanto em relação a tamanho quanto em tempo e, para não dar erro com o "*mimread*", utiliza-se esse artifício.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML


def display_side_by_side(source, driving, generated=None):
    fig = plt.figure(figsize=(8 + 4 * (generated is not None), 6))

    ims = []
    for i in tqdm(range(len(driving)), "generating slides"):
        cols = [source]
        cols.append(driving[i])
        if generated:
        # if generated is not None:
            cols.append(generated[i])
        im = plt.imshow(np.concatenate(cols, axis=1), animated=True)
        plt.axis('off')
        ims.append([im])

    ani = animation.ArtistAnimation(fig, ims, interval=50, repeat=False)
    plt.close()
    return HTML(ani.to_html5_video())


display_side_by_side(source_image, driving_video)
```

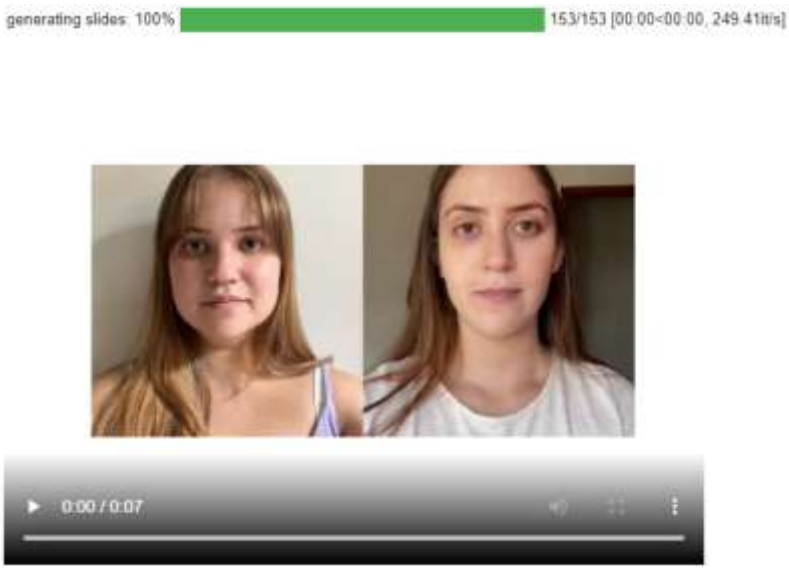

Figura 6 - Foto e vídeo redimensionados e colocados no mesmo frame. Fonte: Próprio Autor

São utilizadas diversas bibliotecas como *"numpy"*, que fornece um grande conjunto de funções e operações que ajudam programadores a executar facilmente cálculos numéricos; *"matplotlib"*, biblioteca de software para criação de gráficos e visualizações de dados em geral; e *"HTML"*, que refere-se a uma linguagem de computador que compõe a maior parte das

páginas da internet e dos aplicativos online composta por uma série de marcações que dizem para os servidores da web qual é o estilo e a estrutura de um documento.

Dando continuidade, é criada uma função de *display*, que recebe a imagem e o vídeo, onde para cada um dos frames do vídeo, a imagem será concatenada (tudo isso definido dentro do "*for*"). Ou seja, nesse ponto concatena imagens e vai regerando o vídeo. No fim, retorna *html*.

Finalmente, o modelo é carregado e o colocado para animar. Ele usa a demo e o modelo de rede neural que foi carregada como explicado anteriormente, disponibilizada em [29] para gerar a *deepfake*, através do carregamento de *checkpoints* previamente testados e preparados em [2]. Essa animação agora também será mostrada no display, com, então, 3 argumentos, *"display_side_by_side",* concatenados.

```
from demo import load_checkpoints
generator, kp_detector = load_checkpoints(config_path='config/vox-256.yaml',
                            checkpoint_path='/content/gdrive/My Drive/deepfake/vox-cpk.pth.tar')
```

```
from demo import make_animation
from skimage import img_as_ubyte
from skimage.io import concatenate_images

predictions = make_animation(source_image, driving_video, generator, kp_detector, relative=True)

display_side_by_side(source_image, driving_video, predictions)
```

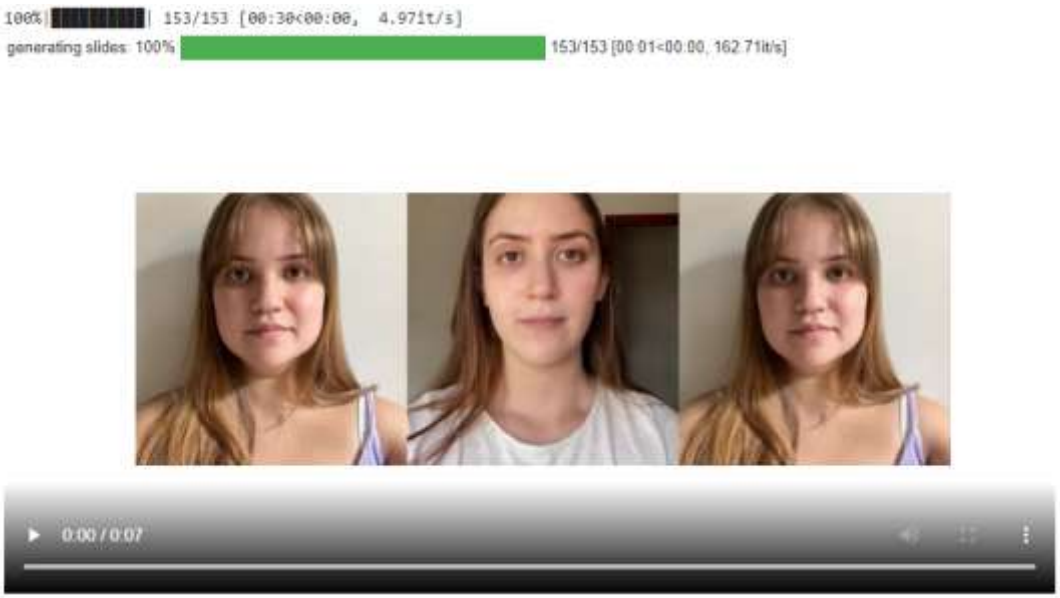

Figura 7 - Processo de criação da *deepfake*. Fonte: Próprio Autor

Nesse ponto, já com a imagem animada disponível, é necessário editar apenas mais um detalhe, a velocidade do vídeo. Para isso, são carregados os dois arquivos usando *"moviepy"*, que são o vídeo original e o gerado, pega-se a duração, divide-se a duração de um para o outro para saber onde deve-se alterar/atrasar, no vídeo final. Isto é, ao realizar essa ação de divisão, na verdade está sendo mapeado o espaço de uma dimensão em outro espaço de outra dimensão. Através do *"ffmpeg"* esse vídeo será lido e em seguida passará um filtro que servirá para configurar os pontos, com *"setpts"*. Infelizmente, com isso perde-se um pouco de resolução.

```
from moviepy.editor import VideoFileClip
generated_duration = VideoFileClip("../generated.mp4").duration
original_duration = VideoFileClip(movie_source).duration
multiple = generated_duration / original_duration
print( generated_duration, original_duration, multiple )
```

```
setpts = f'setpts=PTS/{multiple}'
!ffmpeg -y -hide_banner -loglevel warning -i "../generated.mp4" -filter:v $setpts "../generated_speed_control.mp4"
print(f"Colocando o áudio")

final_name = f"../generated_final_{base_name}.mp4"
!ffmpeg -y -hide_banner -loglevel warning -i "../generated_speed_control.mp4" -i $generated_audio -c:v copy -c:a aac $final_name

ipython_display(final_name)
```

Por fim, nesse novo vídeo com a velocidade alterada, adiciona-se o arquivo de áudio e com isso é disponibilizado o resultado final para análise. Esse processo foi realizado inúmeras vezes para a obtenção de um banco de dados composto por 10 vídeos.

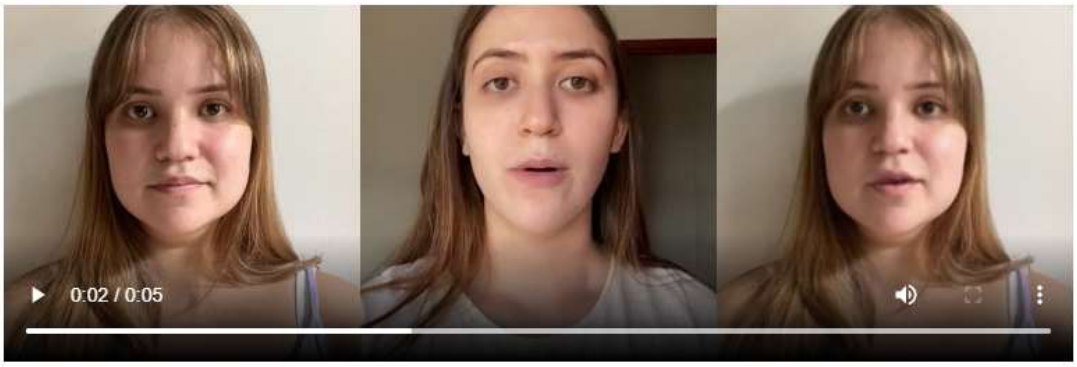

Figura 8 - Deepfake final. Fonte: Próprio Autor

3.3 Análise subjetiva

Para a análise subjetiva foi criado um Formulário do Google, que foi disponibilizado para dois grupos de pessoas. O primeiro, formado por 11 pessoas que não conheciam nem a voluntária da foto e nem a do vídeo base; e o segundo, formado por 5 pessoas que conheciam pelo menos uma delas.

A janela resultante do vídeo ficou formada de três partes, da esquerda para direita: imagem original, vídeo base e imagem animada (*deepfake*). Essa composição teve como objetivo preservar a qualidade. À vista disso, foi especificado aos participantes da pesquisa que analisassem especialmente o *deepfake*. Essa análise foi feita através de duas perguntas formuladas em cima da métrica MOS. Ou seja, tratou-se de uma avaliação numérica acerca da percepcção dos participantes da pesquisa com relação a veracidade e qualidade dos vídeos, disponibilizados através da plataforma *Youtube*. No caso, foram criadas duas perguntas, com 5 opções de resposta, para cada vídeo assim como mostrado na Figura 9 abaixo.

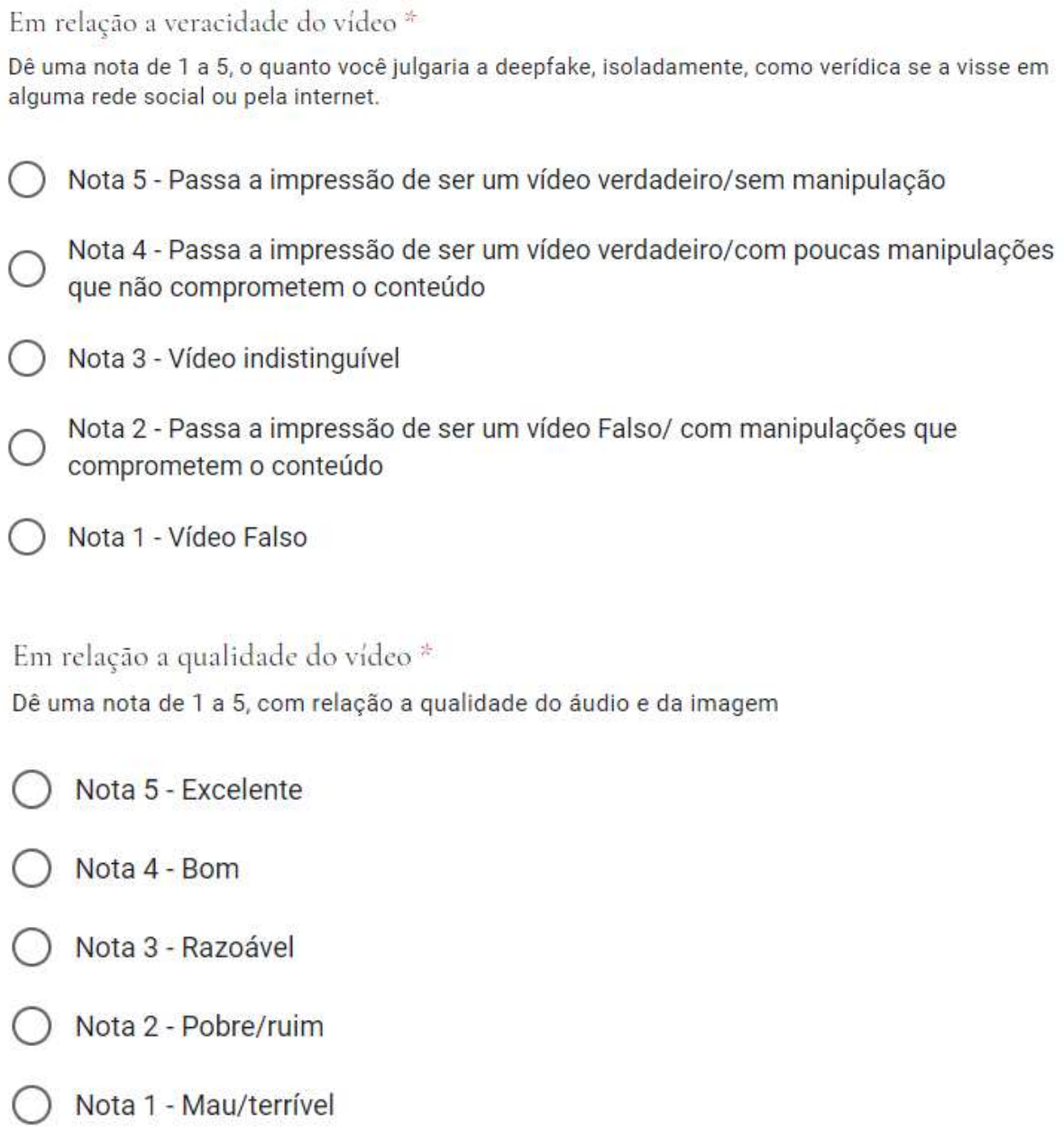

Figura 9 - Perguntas do Formulário. Fonte: Próprio Autor

Para realizar a análise final, foi feita uma média ponderada das notas e no final obtido a média geral para cada vídeo, em relação a veracidade e em relação a qualidade.

3.4 Análise objetiva

Para a análise objetiva, foram extraídos os frames de cada vídeo, através do aplicativo *VLC media player*. Após isso, cada frame foi analisado através de um código, implementado em *Python* que realiza o cálculo do PNSR e exibe esse valor no display.

```python
from math import log10, sqrt
import cv2
import numpy as np

def PSNR(original, compressed):
  mse = np.mean((original - compressed) ** 2)
  if(mse == 0): # MSE igual a zero significa que não há ruído no sinal.
        # Logo, PSNR hnão tem importância nesse caso.
    return 100
  max_pixel = 255.0
  psnr = 20 * log10(max_pixel / sqrt(mse))
  return psnr

def main():
  original = cv2.imread("C:\\Users\\Usuario\\PycharmProjects\\AnaliseObjetiva\\frame0.jpg")
  compressed = cv2.imread("C:\\Users\\Usuario\\PycharmProjects\\AnaliseObjetiva\\frame1.jpg", 1)
  value = PSNR(original, compressed)
  print(f"PSNR value is {value} dB")

if __name__ == "__main__":
  main()
```

O código utiliza a importação de bibliotecas necessárias para seu funcionamento, como *"log10, sqrt"* do módulo *"math"* (no caso esse módulo dá acesso a funções matemáticas, que nesse exemplo são a função logarítmica e a função sqrt que retorna a raiz quadrada de um valor numérico); *"cv2"* que é o nome de importação do módulo para opencv-python, uma biblioteca de programação, de código aberto e inicialmente desenvolvida pela Intel com o objetivo de tornar a visão computacional mais acessível a desenvolvedores; e *"numpy"*.

Observe que para o cálculo do PSNR é necessário que haja uma imagem e sua versão comprimida, para que se possa analisar o MSE. Se esse valor for 0, através de um *"if"* implementa-se uma condicional para retornar o valor 100 para dar continuidade à análise, mesmo que não haja a presença de ruído. As imagens aqui trabalhadas são os frames dos vídeos, ou seja, os frames e suas respectivas versões comprimidas. As duas últimas linhas do código, que compõe o bloco da última condicional, é apenas uma verificação se o programa está sendo executado por si só.

Para a obtenção dos frames a serem trabalhados, foi especificado uma taxa de gravação de imagem, no próprio VLC, de 30. Ou seja, um frame foi gravado a cada 30 frames do mesmo conjunto. Essa escolha de valor foi feita depois de algumas tentativas com valores menores de conjuntos e, ao realizar testes rápidos do PSNR, notou-se que os valores tiverem uma variação praticamente imperceptível. Em razão disso, visando otimizar o trabalho, utilizou-se esse valor de 30 para a taxa, gerando assim 4 frames e, consequentemente, 4 frames comprimidos obtidos

de maneira online, para cada um dos 10 vídeos do banco de dados criado, totalizando um espaço amostral de 80 imagens. Dessa maneira foi possível fazer a análise de tal forma que o PSNR final de cada vídeo ficou definido como a média dos PSNR's encontrados isoladamente para cada frame.

Como a extração dos frames foi realizada diretamente no computador local, foi mais eficaz que, diferentemente do código para criação dos *deepfakes*, rodado no Google Colab, esse código fosse rodado em uma IDE, ou ambiente de desenvolvimento integrado, que nada mais é que um software para criar aplicações que combina ferramentas comuns de desenvolvimento em uma única interface gráfica do usuário. No caso, a IDE utilizada foi o *PyCharm*.

4 Resultados

Depois de obter os resultados do questionário e dos cálculos para o PSNR, foi possível avaliar a qualidade da metodologia proposta.

Com relação a análise subjetiva, foram obtidos os seguintes resultados:

Tabela 3 - Resultados do formulário para o Grupo1

	VÍDEO	MÉDIA EM RELAÇÃO A VERACIDADE	MÉDIA EM RELAÇÃO A QUALIDADE
GRUPO 1	1	3,64	4
	2	3,91	4,18
	3	3,82	4,18
	4	3,27	3,73
	5	3,64	4
	6	3,82	3,82
	7	3,64	3,73
	8	2,64	3,45
	9	4,36	4,1
	10	4,36	4,27
	MÉDIA GERAL	**3,71**	**3,95**

Fonte: Elaborado pelo Autor

Tabela 4 - Resultados do formulário para o Grupo2

	VÍDEO	MÉDIA EM RELAÇÃO A VERACIDADE	MÉDIA EM RELAÇÃO A QUALIDADE
GRUPO 2	1	4	4,4
	2	4	4,6
	3	4,4	4,6
	4	3,6	4,8
	5	3,8	4,6
	6	3,6	4,6
	7	4,6	4,8
	8	3	4,4
	9	4	4,8
	10	4,4	4,8
	MÉDIA GERAL	**3,94**	**4,64**

Fonte: Elaborado pelo Autor

Observe que essas tabelas foram criadas a partir das respostas dos formulários de uma maneira geral. Mas, para realizar uma análise mais minunciosa, temos os gráficos a seguir:

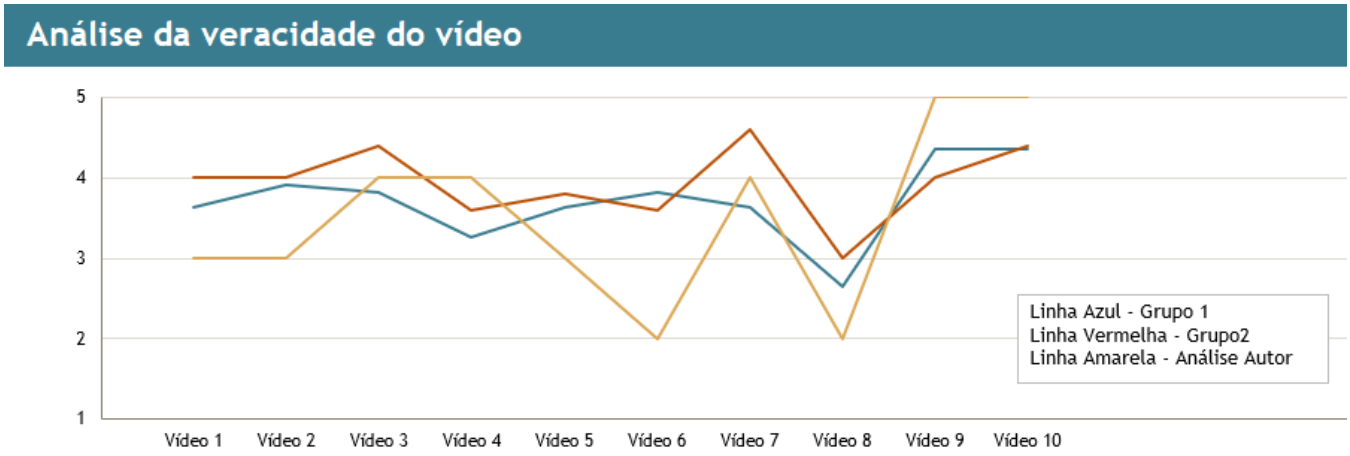

Figura 10 - Gráfico com as notas da veracidade dos vídeos. Fonte: Próprio Autor

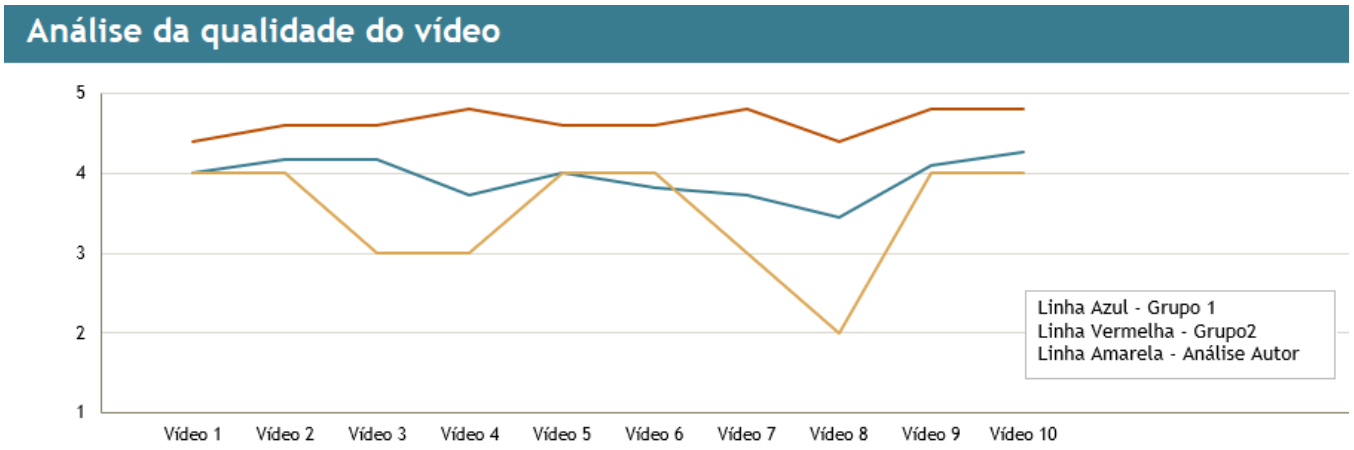

Figura 11 - Gráfico com as notas da qualidade dos vídeos. Fonte: Próprio Autor

Esses gráficos foram obtidos através da avaliação do grupo 1, representado pela cor azul; do grupo 2, representado pela cor vermelha; e, por fim, da análise pessoal do próprio autor, representado pela cor amarela. Esse último levou em consideração adaptações que foram necessárias como luz, enquadramento, ambiente, assim como atenção a detalhes como o movimento da boca, atraso na fala ou aparecimento de dentes falsos, por exemplo. Dessa forma, esses gráficos permitiram realizar uma análise mais detalhada por contar com a participação da opinião de quem criou os *deepfakes*.

Os resultados obtidos foram satisfatórios, uma vez que, como explicado anteriormente algo em torno de 4,3 e 4,5 é considerado um alvo de excelente qualidade enquanto que seria algo inaceitável abaixo de um MOS de aproximadamente 3,5. Logo, para o grupo 1 obtivemos ambos os aspectos (veracidade e qualidade) dentro de algo considerado aceitável e plausível,

enquanto que para o grupo 2, a veracidade ficou razoável/aceitável enquanto que a qualidade já foi considerada excelente. Em relação a avaliação pessoal do próprio autor, também foram obtidos resultados ≥ 3,5, portanto, dentro do aceitável. Diante disso, tem-se que os vídeos manipulados podem se passar como reais de maneira despercebida ou com uma percepcção leve de alteração, ou seja, o mecanismo abordado nesse trabalho, para criação de *deepfakes*, por esse mecanismo subjetivo de avaliação, está dentro do aceitável.

Com relação a análise objetiva, foram obtidos os seguintes resultados:

Tabela 5 - Resultado do PSNR

VÍDEO	PSNR FRAME1	PSNR FRAME2	PSNR FRAME3	PSNR FRAME4	MÉDIA (dB)
1	37.96	38.04	37.72	37.68	37.85
2	37.50	37.60	37.59	37.46	37.54
3	37.86	38.02	37.75	37.81	37.86
4	37.01	37.11	36.88	37.12	37.03
5	37.57	37.26	37.35	37.41	37.40
6	37.71	37.55	37.39	37.53	37.55
7	37.65	37.69	37.80	37.62	37.69
8	37.93	37.85	37.86	37.87	37.88
9	37.80	37.60	37.72	37.73	37.71
10	37.84	37.81	37.69	37.70	37.76

Fonte: Eleborado pelo Autor

De acordo com o apresentado na Tabela 2, todos os resultados obtidos se enquadram como excelentes por estarem acima de 37 dB. Portanto, os vídeos manipulados são

considerados de boa qualidade. Entretanto, é importate ressaltar que essa análise foi feita em cima do vídeo como um todo e não especificamente com o *deepfake* isolado.

5 Conclusão

Os avanços tecnológicos acabam servindo como base para diversas mudanças que ocorrem na sociedade, já que ela precisa se adequar aos problemas causados pelo uso dessa tecnologia. O mesmo tipo de mudanças pode ocorrer com a popularização dos *deepfakes*.

No início, os *deepfakes* exigiam que os usuários tivessem profundo conhecimento em programação. Entretanto, com o tempo, à medida que sua popularidade crescia, as pessoas passaram a criar aplicativos para automatizar todo o processo e com isso seu uso em massa tornou-se mais acessível. Como essa técnica requer recursos avançados para o processamento de redes neurais, dentre outros, a qualidade média dos vídeos acaba não sendo tão boa, mas o suficiente para enganar muita gente, como pôde ser visto nas análises desse projeto. Com uma foto foi possível criar um vídeo manipulado suficientemente convincente. Quer dizer, usando essa tecnologia, pode-se facilmente aplicar golpes em pessoas criando vídeos caluniosos, espalhando mensagens falsas que podem colocar não só a reputação de alguém em jogo, como também sua segurança. Por isso, a disseminação de *deepfakes* é uma das maiores preocupações de governantes, juristas e jornalistas em todo o mundo, pois ela ataca algo que até então era tido como verdade incontestável: a credibilidade de uma gravação em vídeo.

A base de dados criada com sucesso através da implementação do código em Python, permitiu que a análise em cima da sensibilidade humana no discernimento dos *deepfakes* fosse realizada. Dessa forma, com os resultados obtidos utilizando, principalmente, o critério MOS, foi possível perceber que se trata de uma tecnologia capaz de enganar majoritariamente a população exposta a ela. Uma vez que os resultados aqui obtidos levaram em consideração não só uma análise objetiva matemática com o PSNR, mas também, a opinião de grupos compostos por pessoas totalmente externas, que não possuíam um contato prévio com nenhuma das voluntárias para a criação do *deepfake*; e grupos compostos por pessoas que já possuíam algum contato ou conhecimento anterior. Ambos os resultados foram a favor da imagem falsa, isto é, os vídeos manipulados foram considerados como suficientemente convencíveis de serem reais.

Assim, esse trabalho cumpriu com êxito os objetivos propostos levando à conclusão de que, com o crescimento significativo do uso de técnicas digitais para criação de vídeos manipulados com qualidade e a falta de uma tecnologia disponível no mercado para combatê-

las diretamente, inevitavelmente resulta em uma vulnerabilidade não só acerca da percepção, mas sim também na capacidade de julgamento humano sobre essas questões.

Referências Bibliográficas

[1] KIETZMANN, Jan et al. Deepfakes: Trick or treat?. **Business Horizons**, v. 63, n. 2, p. 135-146, 2020.

[2] SIAROHIN, Aliaksandr et al. First order motion model for image animation. **Advances in Neural Information Processing Systems**, v. 32, 2019.

[3] HAYKIN, Simon. **Redes neurais: princípios e prática**. Bookman Editora, 2007.

[4] KOVÁCS, Zsolt László. **Redes neurais artificiais**. Editora Livraria da Fisica, 2002.

[5] COZMAN, Fabio G.; PLONSKI, Guilherme Ary; NERI, Hugo. **Inteligência Artificial: Avanços e Tendências.** Instituto de Estudos Avançados da Universidade de São Paulo, 2021.

[6] EL NAQA, Issam; MURPHY, Martin J. What is machine learning?. In: **machine learning in radiation oncology**. Springer, Cham, 2015. p. 3-11.

[7] SANTOS, Thiago. Google Colab: O que é, Tutorial de Como Usar e Criar Códigos. **Alura**, 2020. Disponível em: < https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar?gclid=CjwKCAiAgvKQBhBbEiwAaPQw3AqZ8ufqOLmPnqfi10vuFNQjbB-wuM8GKm-BfzG3y3a-ArFoUa6hWxoCDh8QAvD_BwE>. Acesso em: 01 de mar. de 2022.

[8] ZISSERMAN, Max Jaderberg Karen Simonyan Andrew; KAVUKCUOGLU, Koray. Spatial Transformer Networks. In: **Nips**. 2015. p. 1-14.

[9] VALENZUELA, Gracieth Mendes. Mecanismo de Seleção de Rede em Ambientes Hetoregêneos Baseado em Qualidade de Experiência (QoE). **Dissertação. Universidade Federal de Pernambuco**. 2011.

[10] ADONIAS, Geoflly L; Ewerton S. Farias, Weslley C. Santos, Carlos Danilo M. Regis. **Análise Objetiva do Número de Bits Menos Significativos em Esteganografia de Imagens Digitais**. 2017.

[11] NAYEM, Khandokar Md; WILLIAMSON, Donald S. Incorporating Embedding Vectors from a Human Mean-Opinion Score Prediction Model for Monaural Speech Enhancement. **Proc. Interspeech** 2021, p. 216-220, 2021.

[12] FURTADO, Maria Inês Vasconcellos. **Redes neurais artificiais: uma abordagem para sala de aula**. Ponta Grossa, PR. Atena Editora, 2019.

[13] ZANIOL, Cristina; PAZINATTO, C. B.; MORAES, J. C. Intervalos de confiança de núcleos de inflação utilizando wavelets e redes neurais. **Anais do X Encontro Regional de Matemática Aplicada e Computacional do Rio Grande do Sul–ERMAC-RS**, 2020.

[14] REDES Neurais. **IBM**, 2020. Disponível em: < https://www.ibm.com/br-pt/cloud/learn/neural-networks >. Acesso em: 23 de mar. de 2022.

[15] SILVA, José Vitor; MATOS, Leonardo. Detecção de pneumonia usando redes neurais convolucionais treinadas com destilação do conhecimento obscuro. In: **Anais da XX Escola Regional de Computação Bahia**, Alagoas e Sergipe. SBC, 2020. p. 51-60.

[16] MATSUMOTO, Daniel Kazuyuki Fugii et al. **Estudo em séries temporais financeiras utilizando redes neurais recorrentes**. Dissertação (Mestrado) – Modelagem Computacional do Conhecimento, Instituto de Computação, Universidade Federal de Alagoas, Maceió, 2019.

[17] LAZZARIN, Lilian NA et al. REDES NEURAIS FEEDFORWARD APLICADAS NA AVALIAÇÃO DO IMPACTO DA POLUIÇÃO ATMOSFÉRICA E VARIÁVEIS CLIMÁTICAS NA SAÚDE HUMANA. In: **Proceedings of the 1st Iberic Conference on Theoretical and Experimental Mechanics and Materials & 11th National Congress on Experimental Mechanics**, Porto/Portugal, 2018. Ed. J.F. Silva Gomes. INEGI/FEUP (2018); ISBN: 978-989-20-8771-9; p. 275-284.

[18] Haykin, S. **Redes neurais: princípios e prática**, Bookman Editora, 2007.

[19] Goodfellow I., Yoshua B., Aaron C**. Deep Learning. The MIT Press**, Cambridge, 2016.

[20] CORVALÁN, Juan Gustavo. Inteligencia Artificial GPT-3, Pretoria y Oráculos Algorítmicos en el Derecho: GPT-3 Artificial Intelligence, Pretoria, and Algorithmic Oracles in Law. **International Journal of Digital Law**, v. 1, n. 1, p. 11-52, 2020.

[21] ROUHIAINEN, Lasse. **Inteligencia artificial**. Madrid: Alienta Editorial, 2018.

[22] JANIESCH, Christian; ZSCHECH, Patrick; HEINRICH, Kai. Machine learning and deep learning. **Electronic Markets**, v. 31, n. 3, p. 685-695, 2021.

[23] DATA Mining Vs Machine Learning Vs Artificial Intelligence Vs Deep Learning. **Softwaretestinghelp,** 2022. Disponível em: < https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/ >. Acesso em: 23 de mar. de 2022.

[24] ARTIFICIAL Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference?. **Sumologic,** 2018. Disponível em: <https://www.sumologic.com/blog/machine-learning-deep-learning/>. Acesso em: 23 de mar. de 2022.

[25] ITU-T Recommendations P.910 P.920 P.930. Subjective video quality assessment methods for multimedia applications, interactive test methods for audiovisual communications, principles of a reference impairment system for video, 1996.

[26] BARATH, D.; HAJDER, L. Novel ways to estimate homography from local affine transformations. In Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP, pages 432- 443, 2016.

[27] SIAROHIN, A., LATHUILIÈRE, S., TULYAKOV, S., RICCI, E., SABE, N. Animating arbitrary objects via deep motion transfer. In CVPR, 2019.

[28] BARBOSA, R. L. et al. Optical flow computation for land-based mobile mappig system images. **Revista Brasileira de Cartografia**, v. 57, n. 2, p. 72-78, aug 2005.

[29] DEEP Fake (passo a passo). **Colab**, 2022. Disponível em: <https://colab.research.google.com/drive/1FRBxmDBSE3-OsyiownBMVWQGsdG9eMIk#scrollTo=I6XlZ-_m2qlm >. Acesso em: 15 de mar. de 2022.

[30] VÍDEO Falso de Obama chama atenção para deep fake news. **Uol**, 2018. Disponível em: <https://www.band.uol.com.br/noticias/jornal-da-band/videos/video-falso-de-obama-chama-atencao-para-deep-fake-news-16430362 >