COLÉGIO TÉCNICO DE CAMPINAS - UNICAMP (COTUCA)
Rua Culto à Ciência, 177 - Centro, Campinas - SP

ÉRIC CARVALHO FIGUEIRA
MARCOS GODINHO FILHO

# Unfake: Desenvolvimento de uma ferramenta para detecção de deepfakes de áudio utilizando inteligência artificial

Orientador: Guilherme de Oliveira Macedo

Início: Fev/2024
Fim: Out/2024

**RESUMO**

A crescente evolução das ferramentas de inteligência artificial (I.A.) as torna cada vez mais eficientes e acessíveis globalmente. No entanto, algumas dessas tecnologias podem ser nocivas, caso usadas de forma mal-intencionada, e isso inclui as *deepfakes*. Elas são um tipo de mídia sintética que gera conteúdos realistas, tendo o potencial de clonar a identidade de um indivíduo, utilizando-a para a propagação de notícias falsas, deterioração de sua reputação e promoção de fraudes e violações de segurança. Assim, são necessárias maneiras de verificar se uma mídia é real ou foi sintetizada artificialmente. No entanto, por mais que existam tecnologias que atendam a essa necessidade, a detecção de *deepfakes* de áudio ainda é um desafio, considerando que ela não é tão efetiva quando se trata de falas na língua portuguesa e possui uma eficácia questionável em áudios com presença de ruídos. Nesse sentido, este projeto tem como objetivo o desenvolvimento de um modelo de I.A. capaz de identificar se um áudio contém uma fala humana ou sintética. Ademais, para a disponibilização desta ferramenta, foi desenvolvido um *website* que permite ao usuário enviar um áudio a uma *Application Programming Interface (API)* codificada, sendo o mesmo processado por uma rede neural classificatória, que retorna uma resposta quanto à autenticidade da fala em questão. Para a elaboração desse software, foram utilizadas bibliotecas da linguagem *Python*, como o *TensorFlow*, para a criação, treinamento e validação da I.A., e o *Flask*, para a API, que pode ser acessada através do website desenvolvido com o framework *front-end Next.js*. Desse modo, possibilita-se a identificação de *deepfakes* por usuários leigos, de maneira robusta e eficaz, contribuindo para um ambiente digital mais seguro e confiável, além de fomentar futuras pesquisas na área com o uso dos dados obtidos no projeto.

**SUMÁRIO**

## INTRODUÇÃO

Nos últimos anos, houve um desenvolvimento de aplicações de mídia social e a adoção generalizada pela população de dispositivos conectados à internet como computadores, smartphones e tablets, revolucionando a comunicação e interação entre os usuários. Rapidamente, os mesmos se tornaram capazes de produzir, compartilhar e interagir com conteúdos de forma simples e imediata, criando uma cultura de auto expressão digital (MUBARAK et al., 2023).

No entanto, esse avanço tecnológico também trouxe consigo novos desafios, como as *deepfakes*. O termo se refere a conteúdos gerados ou manipulados por técnicas de Inteligência Artificial (IA), que diferem de outras ferramentas de edição manual por produzirem vídeos, imagens, áudios e textos muito semelhantes à realidade (KHANJANI; WATSON; JANEJA, 2023).

Tais conteúdos são gerados com uso de diversas redes neurais profundas, como as redes neurais convolucionais (CNN) e recorrentes (RNN), as redes adversárias generativas (GAN), os autoencoders variáveis (VAE) e os modelos de difusão (DMs) (MUBARAK et al., 2023). Vale lembrar que o desempenho dos mesmos depende da quantidade de dados que recebem, e, atualmente, quantidades massivas de dados são produzidas diariamente na internet, tornando-os cada vez mais eficientes.

Apesar das *deepfakes* oferecerem aplicações positivas em várias áreas, como entretenimento, educação e publicidade, elas têm se mostrado uma estratégia eficaz na propagação de notícias falsas, roubo de identidade e fraudes, configurando uma ameaça significativa no âmbito social, político e econômico. A preocupação com o tema se traduz no número de artigos científicos publicados relacionados ao mesmo entre 2017 e 2022, período no qual o valor cresceu de menos de 100 para mais de 3000 (MUBARAK et al., 2023).

Além disso, o surgimento de ferramentas user-friendly para criação de deepfakes possibilitou a fabricação de conteúdo sintético altamente realista entre usuários leigos em seus smartphones e computadores pessoais, agravando ainda mais os efeitos nocivos da tecnologia (MUBARAK et al., 2023).

Um caso que ilustra o perigo da tecnologia ocorreu em 2019: criminosos usaram um software baseado em IA para se passar pelo chefe alemão de um CEO de uma empresa de energia, que recebeu uma ligação onde lhe foi pedida uma transferência de 220.000 euros para um suposto fornecedor húngaro. A transferência foi realizada e os criminosos receberam o dinheiro. Quando o CEO percebeu que tudo era uma fraude, já era tarde demais (STUPP, 2019).

Outro incidente envolvendo prejuízos econômicos aconteceu em junho de 2020, quando um funcionário de uma empresa de tecnologia recebeu uma mensagem de um suposto CEO que também pedia transferência de dinheiro, sendo o mesmo na verdade uma voz criada com uso de IA (PRADO, 2021).

Já em se tratando das consequências políticas das *deepfakes*, um incidente marcante ocorreu nas vésperas das eleições primárias do estado de New Hampshire, Estados Unidos, em 2024. O fato é que um áudio falso simulando a voz do presidente americano Joe Biden circulou nas redes sociais do estado, numa tentativa de influenciar o resultado das eleições, um problema que pode se repetir no Brasil e trazer graves consequências à democracia (MARTINS, 2024).

Assim, considerando os dados levantados, torna-se necessária a criação de ferramentas que possam detectar *deepfakes* e informar aos usuários se o conteúdo consumido é real ou não. No entanto, apesar de haver tecnologias que atendam ao requisito, a maioria delas foca na detecção de imagens e vídeos (MASOOD et al., 2021), deixando em segundo plano as *deepfakes* de áudio.

Desse modo, a nossa pesquisa consiste no desenvolvimento de um modelo de inteligência artificial que detecte deepfakes de áudio, especialmente em português, bem como a disponibilização do mesmo em um website público e acessível a todos.

## OBJETIVOS E RELEVÂNCIA DO TRABALHO

A geração de áudios falsos é possibilitada por duas tecnologias: as ferramentas *Text-To-Speech* (TTS), que convertem um texto em uma fala que imita a voz de uma pessoa real, e as do tipo Voice Conversion (VC), que modificam as características da voz de uma pessoa para se adaptar às do alvo desejado. Entre os principais modelos de geração de *deepfakes* de áudio, estão o Char2Wav, o WaveNet, o WaveGlow, o Tacotron e o MelNet (KHANJANI; WATSON; JANEJA, 2023).

Quanto à detecção, há, atualmente, três principais abordagens na criação de algoritmos que identificam áudios falsos. A primeira é baseada em recursos perceptivos criados manualmente, que envolve a comparação de diferenças estatísticas entre a fala humana e sintética, usando algoritmos avançados de processamento de sinais. A segunda é a detecção genérica baseada em características espectrais, normalmente usando métodos de aprendizado de máquina para extrair características da frequência dos sinais de áudio e compará-los estatisticamente (HONG, 2023).

Por fim, a última é a baseada em *deep learning*, ou aprendizagem profunda, que, segundo Goodfellow, Bengio e Courville (2016), pode ser definida como:

> [...] um tipo específico de aprendizagem automática que atinge grande poder e flexibilidade ao representar o mundo como uma hierarquia aninhada de conceitos, com cada conceito definido em relação a conceitos mais simples e representações mais abstratas calculadas em termos de representações menos abstratas (tradução nossa).

Essa abordagem, no geral, requer uma grande quantidade de dados classificados em reais ou falsos para obter um bom resultado, e subdivide-se em duas categorias: as baseadas na entrada de áudio bruto, que operam diretamente no áudio em sua forma de onda, e as baseadas em visão computacional (HONG, 2023).

Esta última segue a seguinte metodologia: cada áudio é, primeiramente, pré-processado, sendo transformado em espectrogramas, para em seguida ser fornecido a um modelo, que é treinado para discernir os áudios e dizer se são ou não *deepfakes* (ALMUTAIRI; ELGIBREEN, 2022). Os espectrogramas citados são

representações visuais dos áudios, onde a cor e os eixos x e y representam a amplitude, o tempo e a frequência, respectivamente. Vale lembrar que essa categoria considera não só o conteúdo do áudio mas também a sua representação visual, aumentando a precisão da detecção (HONG, 2023). Entre os diferentes tipos de espectrograma, o constant-Q transform (CQT) é, no geral, mais exato quando utilizado para o *deepfake* de áudio (MÜLLER et al., 2022).

Figura 1 - Métodos de detecção de *deepfake* de áudio baseados em recursos de aprendizagem profunda

| Source | Method | Dataset | Performance | Limitation |
|---|---|---|---|---|
| Lai et al. [212] | Dilated Residual Network and Attentive Filtering Networks AFN | ASVspoof 2017 | EER = 0.089 | For replay attack detection only- Performance depends on the type of nonlinear activation function. |
| Gomez-Alanis et al. [213] | Light Convolutional Gated Recurrent Neural Network (LC-GRNN) | ASVspoof 2015-2017-2019 | EER = 0.0, t-DCF = 0.061 | It needs to be tested against varying conditions, such as different audio qualities and increasingly sophisticated generation techniques. |
| Gomez-Alanis et al. [223] | Gated Recurrent Convolutional Neural Networks (GRCNNs) | ASVspoof 2015-2017-2019 | EER = 0.01, t-DCF = 0.02 | It needs to be tested against varying conditions, such as different audio qualities and increasingly sophisticated generation techniques. |
| Wang et al. [192] | DNN - neuron behaviours | FoR, Sprocket-VC, MC-TTS | Accuracy = 99% | Degraded performance against adversarial noise attacks |
| Subramani & Rao. [214] | EfficientCNN and RES-EfficientCNN | ASVSpoof2019, RTVCSpoof | macro-F1 = 97 | Need to be tested against varying conditions |
| Ma et al. [215] | CNN -Resnet34 - Deep Residual Learning based on ResNet 189 | ASVspoof2019 | EER = 0.03, t-DCF = 0.08 | Complex architecture |
| Zhang et al. [216] | ResNet 189 with Transformer Encoder | FoR, ASVspoof2019 | EER = 3.99% | Complex architecture |
| Tak et al. [218] | CNN based on RawNet2 [217] | ASVspoof 2019 | EER = 3.5%, t-DCF = 0.904 | Need to be tested against varying conditions and datasets |
| Zhang et al. [219] | ResNet [194] with SGD | ASVspoof 2019 | EER = 0.02 | The model is sensitive to parameter tuning and faces challenges when dealing with complex real-world data. |
| Conti et al. [220] | 3D-CRNN with random forest, Speech Emotion Recognition | ASVspoof 2019, LibriSpeech, LJSpeech, IEMOCAP | Accuracy = up to 100% | The study does not fully explore its effectiveness across different emotional ranges or languages. |
| Mo et al. [224] | CNN - Resnet34 - Autoencoder, OOD data | ASVspoof 2019 | EER = 0.01, t-DCF = 0.036 | Need to be tested against varying conditions and datasets |
| Papastergiopoulos et al. [222] | 2D-CNN - VGG16 | FoR, LJSpeech, VoxForge, TIMIT | Accuracy = 93% | Drop in performance with dissimilarities between datasets |
| Salvi et al. [221] | CNN Based on RawNet2 [217], Audio folding | ASVspoof 2019 | Accuracy = 0.95 | It needs to be tested against varying conditions, such as different audio qualities and increasingly sophisticated generation techniques. |

Fonte: A Survey on the Detection and Impacts of Deepfakes in Visual, Audio, and Textual Formats.[1]

[1] Disponível em: <https://ieeexplore.ieee.org/document/10365143?denied=>.

Por fim, foram desenvolvidos vários modelos que usam a última estratégia para detectar áudios falsos (ver Figura 1). No entanto, notam-se limitações em todos eles, como desempenho dependente do tipo de função de ativação não linear, necessidade de ser testado em condições variáveis, tais como diferentes qualidades de áudio e técnicas de geração mais sofisticadas, eficácia reduzida contra ataques de ruído adversários e em casos de bases de dados muito distintas, e arquiteturas complexas demais (MUBARAK et al., 2023).

Como apontam Almutairi e Elgibreen (2022), quase todos os métodos de detecção de *deepfakes* de áudio existentes foram treinados para a língua inglesa, com poucas exceções que não incluem um específico para o português, de modo que os falantes dessa língua, incluindo os brasileiros, estão mais sujeitos aos impactos dessa tecnologia.

Outrossim, percebe-se uma avaliação limitada de falas com ruídos e sons de ambiente nos métodos de detecção existentes, levando a uma baixa robustez que permite a atacantes cibernéticos enganarem os modelos ao introduzir sons do mundo real (ALMUTAIRI; ELGIBREEN, 2022) ou realizar em tais áudios alterações superficiais, difíceis de detectar por um ser humano (KAWA, 2023).

Desse modo, a pesquisa tem por objetivo geral desenvolver um modelo classificatório de inteligência artificial utilizando *deep learning* que seja capaz de identificar, com precisão e eficiência, potenciais áudios produzidos com *deepfake* em português, sendo esse modelo disponibilizado em um website, permitindo aos usuários fazer upload de um áudio e receberem a probabilidade do conteúdo em questão ser real ou sintético.

Já de um ponto de vista específico, a pesquisa possui os seguintes objetivos:

- Descobrir bancos de dados públicos de falas em português com suas respectivas transcrições;
- Identificar modelos existentes para geração de *deepfakes*, tanto do tipo *Text-To-Speech* quanto *Voice Conversion*;
- Alimentar os modelos identificados com os áudios para gerar *deepfakes* e gerar uma base de dados com os mesmos;

- Realizar o pré-processamento dos áudios reais e sintéticos com a produção de espectrogramas;
- Manipular os espectrogramas com a introdução de ruídos e sons de ambiente em uma amostra dos mesmos;
- Projetar as possíveis arquiteturas da rede neural classificatória;
- Codificar e testar os modelos concebidos;
- Treinar os modelos com uma amostra da nossa base de dados;
- Avaliar e comparar a performance de cada modelo, elegendo o mais adequado para o produto final;
- Terminar o treinamento da arquitetura escolhida, fazendo ajustes conforme o necessário;
- Demonstrar que o modelo é eficaz para a nossa base de dados;
- Desenvolver uma API que seja capaz de receber um áudio, convertê-lo em espectrograma, fornecer a conversão para o modelo, receber um output do mesmo indicando se o áudio é uma deepfake ou não e enviar de volta essa resposta.
- Definir o design do aplicativo, com esboços de telas e funcionalidades;
- Construir o website de modo que ele consiga obter um áudio fornecido pelo usuário, enviá-lo para a API, receber uma resposta quanto à sua veracidade e exibir essa informação ao usuário.

O trabalho desenvolvido é de alta relevância social, política e econômica, já que pode auxiliar a mitigar impactos nos três setores. Ademais, tendo em vista a não existência de um conjunto de áudios reais e falsos em português, sua criação juntamente com um modelo classificatório que leve em consideração áudios com ruídos pode servir de base para pesquisas futuras na área. Por fim, ao se produzir um modelo específico para a língua portuguesa, a sociedade, o governo, e empresas e instituições brasileiras tornam-se aptas a distinguir os conteúdos produzidos nas redes, levando a uma menor chance de serem enganadas, mal-informadas e prejudicadas, e contribuindo para um ambiente digital mais seguro e confiável.

## DESENVOLVIMENTO DO PROJETO

A pesquisa desenvolvida seguiu o método de engenharia, ou seja, identificado o problema, buscou-se propor uma solução para o mesmo através de um produto. Além disso, ela se tratou de uma pesquisa aplicada, isto é, cuja produção de conhecimento foi dirigida à solução da questão escolhida.

No início do projeto, foram discutidas as possíveis ideias para o seu desenvolvimento, que envolveram temas como educação, bem estar digital, fake news e acessibilidade. Dois temas finais foram considerados: um aplicativo para o ensino de libras usando reconhecimento de imagem e um aplicativo para detectar *deepfakes* de áudio. A ideia escolhida foi a última, pela relevância e interesse pelo assunto.

Em seguida, realizou-se a pesquisa teórica do projeto, buscando-se fontes que indicassem a sua importância, como casos em que *deepfakes* trouxeram malefícios à sociedade, economia e política. Também começou-se a estudar metodologias para detectar *deepfakes*, com fundamentação baseada, quase inteiramente, no artigo *Uncovering the Real Voice: How to Detect and Verify Audio Deepfakes*, de Tan Jian Hong.

Considerando os dados presentes no artigo, decidiu-se que a metodologia empregada seria o uso de *deep learning* e, especialmente, visão computacional, para a detecção das *deepfakes*, pelas numerosas pesquisas que utilizaram a abordagem e atingiram resultados promissores.

Assim, seria necessário criar um modelo classificatório que, alimentado com dados sobre áudios reais e sintéticos, fosse capaz de classificá-los de acordo com sua respectiva categoria. Dessa forma, foi realizado o estudo sobre inteligência artificial, geração e identificação de *deepfakes* e o curso *Intro to Deep Learning with TensorFlow*, da plataforma *Codecademy* foi cursado, onde foram aprendidos conceitos e práticas essenciais para a estruturação, treinamento e validação de redes neurais.

Ademais, foi tomada a iniciativa de codificar um website em *Next.j*s e uma API em *Flask* para permitir o uso, por usuários leigos, do modelo a ser desenvolvido no projeto, expandindo o acesso ao mesmo.

Para atingir os objetivos da pesquisa, seria necessário um *dataset* de áudios reais e sintéticos, que seria usado para o treinamento do modelo a ser desenvolvido. Assim, foi feita uma pesquisa de *datasets* já existentes elegíveis para esse fim, como o "ASVspoof 2019" (YAMAGISHI et al., 2019), contendo *deepfakes* criadas a partir de falas em inglês. Contudo, percebeu-se que não havia *dataset* específico para a língua portuguesa. Isso foi confirmado pelos artigos *A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions*, de Almutairi e Elgibreen, e *Audio deepfakes: A survey*, de Khanjani, Watson e Janeja, o que levou à decisão de criar um *Dataset* inovador em português, permitindo a detecção de *deepfakes* na língua falada em território brasileiro.

À vista disso, a pesquisa consistiu, inicialmente, em encontrar um grande conjunto de áudios dos quais as vozes poderiam ser clonadas através de ferramentas de *Text-to-Speech* (TTS) ou *Voice Conversion* (VC), visando a produção de *deepfakes*.

Um desses bancos encontrado foi o *Corpus* do Centro de Estudos em Telecomunicações da PUC-Rio (CETUC) (OLIVEIRA, 2012), disponibilizado através do "Fala Brasil", um grupo de pesquisa em processamento de fala em português brasileiro da Universidade Federal do Pará. O dataset é composto por áudios de cerca de 5 segundos com 1.000 sentenças, gravados por 101 locutores dos sexos masculino e feminino, totalizando aproximadamente 143 horas de áudio. Essas gravações, no formato *wav*, são acompanhadas de suas transcrições no formato *txt*.

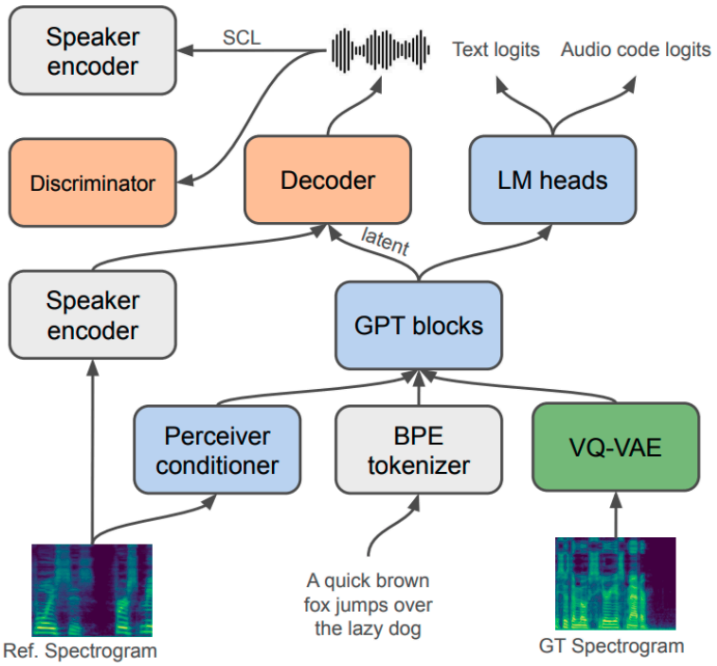
Outro conjunto de dados encontrado foi o "*Mozilla Common Voice"* (ARDILA et al., 2019), que possui milhares de áudios agrupados juntamente com seus conteúdos em arquivos no formato *csv.* Seu lado negativo, porém, é a não diversidade de vozes, apesar da quantidade de horas em áudio disponibilizadas, que é muito elevada.

Com ambos os conjuntos de dados (áudios e transcrições) em mãos, iniciou-se a etapa de síntese de *deepfakes*. A primeira fase dessa etapa foi analisar as redes neurais existentes para esse fim. Entre os distintos modelos encontrados, destacam-se: *Tacotron 2* (SHEN et al., 2017), *Hifi-GAN* (KONG; KIM, J; BAE, 2020), *GlowTTS* (KIM et al., 2020), *WaveNet* (VAN DEN OORD et al., 2016), entre outros.

Contudo, a maioria deles apresentou problemas de instalação de pacotes ou exigiu componentes gráficos avançados e tempo de execução muito acima do disponível. Além disso, eles não haviam sido treinados para a língua portuguesa, tornando-os impraticáveis para nossos objetivos.

Todavia, foi descoberto um modelo *Text-To-Speech* chamado "XTTS" (CASANOVA et al., 2024) que se destacou por ter sido treinado previamente em diferentes línguas, incluindo o português, e entregou resultados aceitáveis, além de ser *open-source*, isto é, com os códigos públicos. A arquitetura do modelo pode ser observada na Figura 2. Por falta de opções e recursos, ele foi utilizado para a produção do nosso *Dataset*, apesar de entendermos que, ao utilizar um só modelo, o conjunto de dados ficou limitado.

Figura 2: Arquitetura do modelo XTTS




Fonte: XTTS: a Massively Multilingual Zero-Shot Text-to-Speech Model[2]

[2] Disponível em: <https://arxiv.org/pdf/2406.04904>.

Encontrado o modelo, foi decidido alimentá-lo com dados do *Corpus* do CETUC, pois ele contém gravações curtas, de até 10 segundos, somadas às suas transcrições, exatamente o exigido pela rede neural.

Assim, foi escrito um programa na linguagem de programação *Python* que, dada uma pasta contendo áudios e transcrições de uma das vozes, fornece esses dados para a rede, que então, através dos padrões de fala e das transcrições dos áudios, gera as *deepfakes*. O código foi feito baseado na documentação da biblioteca “Coqui-TTS” (GÖLGE et al., 2022), uma solução eficiente que possui funções prontas para geração de *deepfakes* através de modelos *Text-To-Speech,* permitindo a alimentação dos mesmos com vozes e textos customizados em um tempo de execução aceitável.

Em seguida, foi iniciada a produção do nosso *Dataset*, gerando, para cada voz do *Corpus* disponibilizado pelo grupo “Fala Brasil”, uma *deepfake* com o mesmo conteúdo contido nos arquivos texto do *Corpus*, isto é, das falas originais.

Após a produção dos áudios sintéticos, ocorreu a geração de espectrogramas do tipo *constant-Q transform* (CQT) (SCHÖRKHUBER; KLAPURI; 2010) das *deepfakes* e dos áudios do *Corpus* do CETUC, em escala cinza. Optou-se por essa abordagem pois, como já discutido, ela é mais eficaz para essa forma de detecção. Para a obtenção desses dados, foram empregadas bibliotecas na linguagem *Python*, tais como *Librosa* e *Matplotlib*, que permitem a manipulação desse tipo de conteúdo, com métodos para geração de espectrogramas CQT, redimensionamento das imagens e visualização das mesmas.

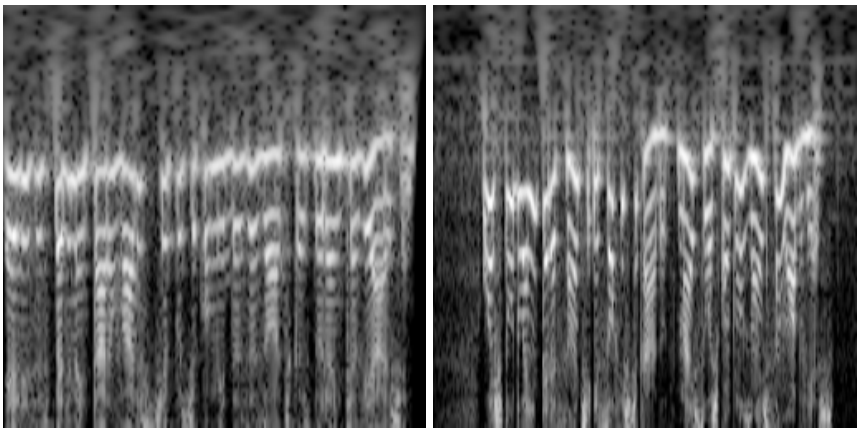
A formulação matemática envolvida na geração dos espectrogramas pode ser observada na Figura 3, que descreve a transformação CQT de um sinal discreto no domínio do tempo x(n). Já exemplos de imagens geradas podem ser vistos na Figura 4, onde, à direita, observa-se um espectrograma de uma *deepfake*, e, à esquerda, de uma fala real. A principal diferença da transformação CQT para outros tipos de espectrogramas é que ela é capaz de representar diferentes notas e escalas musicais.

Figura 3: Transformação CQT

$$X^{\mathrm{CQ}}(k,n) = \sum_{j=n-\lfloor N_k/2 \rfloor}^{n+\lfloor N_k/2 \rfloor} x(j)a_k^*(j-n+N_k/2)$$

Fonte: Constant-Q transform toolbox for music processing[3]

Figura 4: Versão inicial dos espectrogramas reais e falsos



Fonte: Imagem dos autores

A próxima etapa consistiu no processamento do *Dataset*. Vale lembrar que atacantes cibernéticos introduzem pequenas alterações - ruídos - nos áudios, burlando tais sistemas, fazendo com que a robustez dos modelos de detecção existentes acabe sendo prejudicada.

Assim, foi desenvolvido um programa para adicionar sons ambientes nos áudios, e ruídos nos espectrogramas, sendo realizados esses dois procedimentos, bem como uma combinação dos dois processos. Com esta abordagem, é maximizada a diversidade de dados e minimizada a chance de ocorrer o enviesamento do modelo.

Dessa forma, foram feitas tais alterações, para cada áudio, escolhendo proporções de maneira arbitrária, como pode-se observar abaixo:

- 100% dos áudios foram convertidos para espectrogramas e não sofreram nenhum tipo de modificação;

[3] Disponível em: <https://core.ac.uk/download/pdf/144846462.pdf>.

- 16,7% dos áudios tiveram sons ambiente adicionados e então foram convertidos para espectrogramas;
- 16,7% dos áudios foram convertidos para espectrogramas e tiveram ruídos adicionados;
- 16,7% dos áudios tiveram sons ambiente adicionados e então foram convertidos para espectrogramas, que terão também ruídos adicionados.

A introdução de ruídos ocorreu através da extração de amostras de uma distribuição normal parametrizada no formato (tamanho) dos espectrogramas, com desvio padrão de 0,5 e valor esperado valendo 0. Essas amostras foram, então, sobrepostas aos espectrogramas, criando ruídos nos mesmos. A fórmula da distribuição normal pode ser observada na Figura 5, onde σ é o desvio padrão e μ é o valor esperado.

Figura 5: Distribuição normal

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Fonte: Distribuição normal[4]

Já os sons de ambiente citados foram pegos da base de dados “ESC: Dataset for environmental sound classification” (PICZAK, 2015), que contém sons de animais, paisagens naturais, e ambientes doméstico e urbano, sendo estes sobrepostos aos áudios originais utilizando a biblioteca *pydub* em Python.

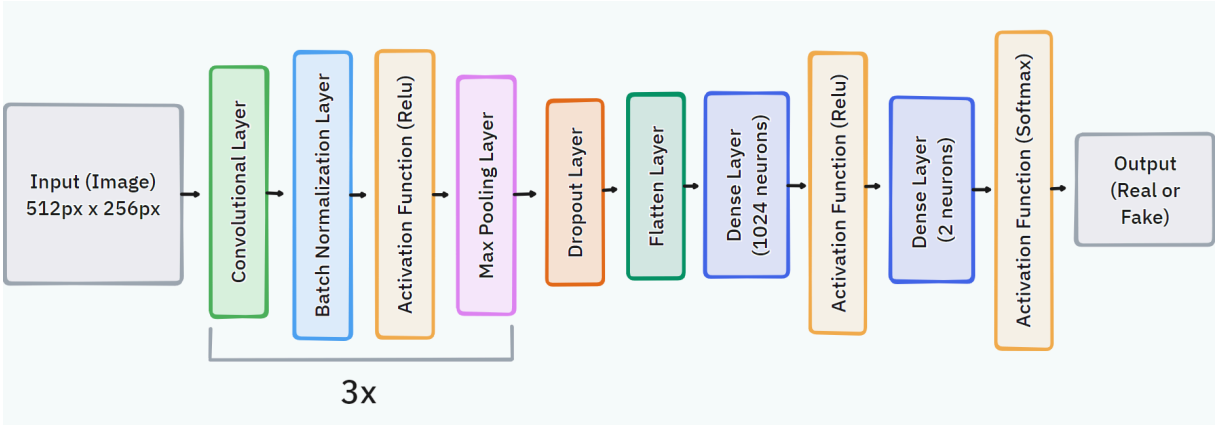
Então, foi desenvolvido o foco do projeto: um modelo de inteligência artificial capaz de identificar áudios sintéticos. Após realizar pesquisas a respeito do tema, optou-se por utilizar visão computacional para esse fim, por ela considerar não só o conteúdo do áudio mas também a sua representação visual, aumentando a precisão da detecção. Assim, foi vital identificar arquiteturas de redes neurais aptas para a realização da tarefa.

[4] Disponível em: <https://pt.wikipedia.org/wiki/Distribui%C3%A7%C3%A3o_normal>.

Realizou-se uma revisão bibliográfica dos métodos utilizados até então, e selecionou-se alguns para testar e comparar seus desempenhos e capacidade de classificação, escolhendo os que melhor se encaixaram na resolução do problema e entregarem resultados satisfatórios. Três arquiteturas principais foram encontradas e possuem tal habilidade: a rede neural convolucional (CNN) (LECUN, 1989), a rede adversária generativa com classificador externo (EC-GAN) (HAQUE, 2021) e o Vision Transformer (DOSOVITSKIY, 2020).

Com respeito à CNN, esta é uma arquitetura de rede neural que utiliza camadas convolucionais e filtros a fim de perceber padrões em imagens e assim poder classificá-las (FUKUSHIMA, 1980). A arquitetura da CNN usada no projeto pode ser observada na Figura 6.

Figura 6 - Arquitetura da rede neural convolucional (CNN) desenvolvida

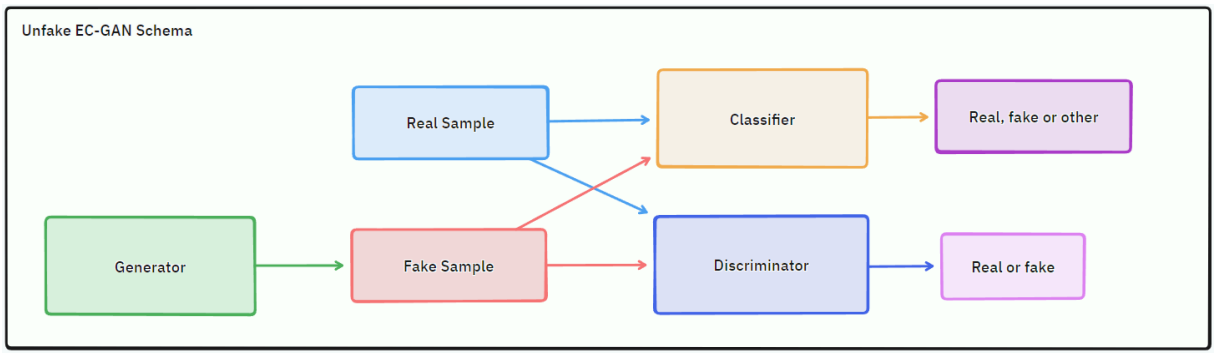



Fonte: Imagem dos autores

Cabe destacar que foram utilizadas técnicas de normalização e regularização na construção do modelo, com introdução de camadas de *Batch Normalization* e *Dropout*. A ideia principal do *Dropout* é eliminar aleatoriamente nós (juntamente com suas conexões) de uma rede neural durante o treinamento, evitando que eles se adaptem demais (SRIVASTAVA, 2013). Já o *Batch Normalization* permite usar taxas de aprendizado muito mais altas e ter menos cuidado com a inicialização. Ela também atua como um regularizador, em alguns casos eliminando a necessidade de *Dropout* (IOFFE, 2015).

Já em relação à EC-GAN, esta é uma adaptação da rede neural adversária generativa (GAN). Deixa-se claro, aqui, que as GANs não foram concebidas para classificar dados, mas sim a fim de gerar novos e parecidos com os originais. Por isso, Haque (2023) propôs a introdução de um classificador externo ao modelo, daí o nome External Classifier GAN. A estrutura e relação entre os diferentes modelos dessa abordagem pode ser observada na Figura 7.

Figura 7 - Arquitetura da rede adversária generativa com classificador externo (EC-GAN)




Fonte: Imagem dos autores

O Vision Transformer, por sua vez, é uma arquitetura de aprendizagem profunda que divide uma imagem de entrada em uma série de patches (em vez de dividir o texto em tokens, como em um Transformer tradicional), serializa cada patch em um vetor e o mapeia para uma dimensão menor com uma única multiplicação de matriz. Ele foi criado como uma alternativa à CNN, no contexto de classificação de imagens.

Tendo, então, as possíveis arquiteturas para a solução do problema, elas foram implementadas em *Python* utilizando a biblioteca *TensorFlow* e treinamentos iniciais foram realizados a fim de testar a entrega de resultados pelas mesmas. Assim, foi feita uma comparação com base numa análise qualitativa quantitativa da performance de cada arquitetura, levando em consideração parâmetros como precisão, taxa de erro e tempo de execução.

Entre as diferentes arquiteturas, a CNN foi a única cujo desempenho foi adequado, com tempo de execução e precisão eficientes para o conjunto de dados construído. Enquanto a GAN, por envolver três modelos distintos e um código complexo, levou um tempo de execução acima do esperado, o Vision Transformer

consumiu memória excessiva, além da capacidade de processamento disponível. Além disso, o Vision Transformer necessitaria de quantidades massivas de dados, muito maiores do que o *Dataset* da pesquisa.

Então, foi feito um treinamento primário da CNN utilizando uma pequena amostra do conjunto de áudios, contendo espectrogramas de 5 vozes reais, rotuladas como *real*, em conjunto com suas *deepfakes* correspondentes, rotuladas como *fake*, além de espectrogramas de sons não humanos aleatórios, rotulados como *other*, totalizando cerca de 12.000 espectrogramas. Aqui, cabe destacar que, na versão inicial do modelo, ele possuía três saídas possíveis, incluindo sons aleatórios, o que foi retirado posteriormente por esse não ser o foco do projeto. O treinamento realizado é descrito a seguir:

- Ele foi realizado por 25 iterações, em um computador com processador 13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz, e memória de 16,0 GB.
- Os hiperparâmetros usados podem ser visualizados na Tabela 1 (são encontradas mais informações sobre eles no Anexo B).

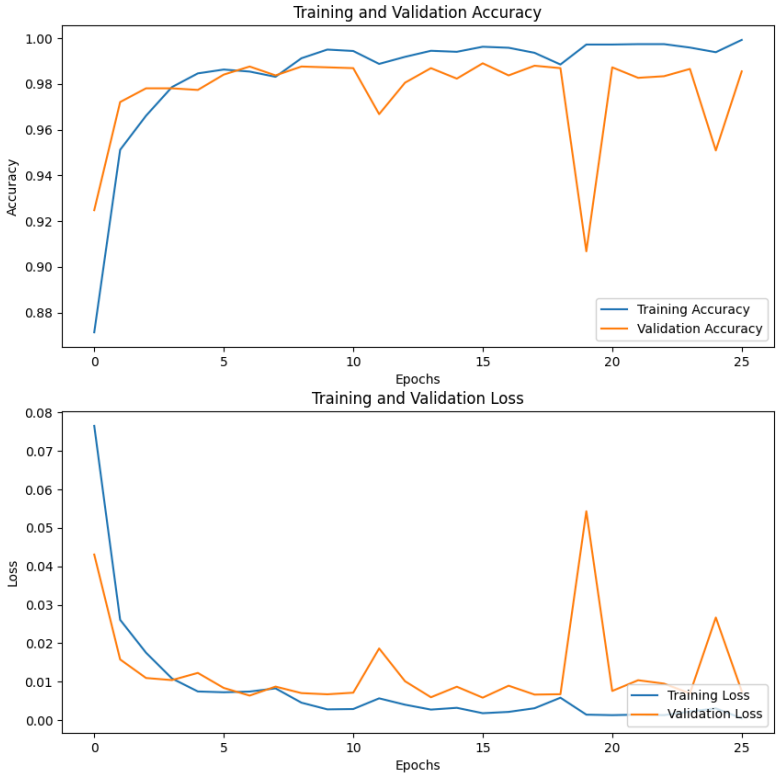
Tabela 1 - Hiperparâmetros e seus respectivos valores utilizados no treinamento inicial da CNN desenvolvida

| hiperparâmetro | valor utilizado |
|---|---|
| número de *epochs* | 100 |
| tamanho do *batch* | 32 |
| taxa de aprendizagem | 0.01 |
| paciência | 15 iterações |
| função de *loss* | *Categorical Cross Entropy* |
| optimizador | *Adam* |

Fonte: Tabela dos autores

Os resultados podem ser visualizados na Figura 8, que possui dois gráficos: um superior e um inferior, que mostram, respectivamente, a precisão e o erro do modelo ao longo das iterações de treinamento.

Figura 8 - Treinamento inicial da CNN desenvolvida




Fonte: Imagem dos autores

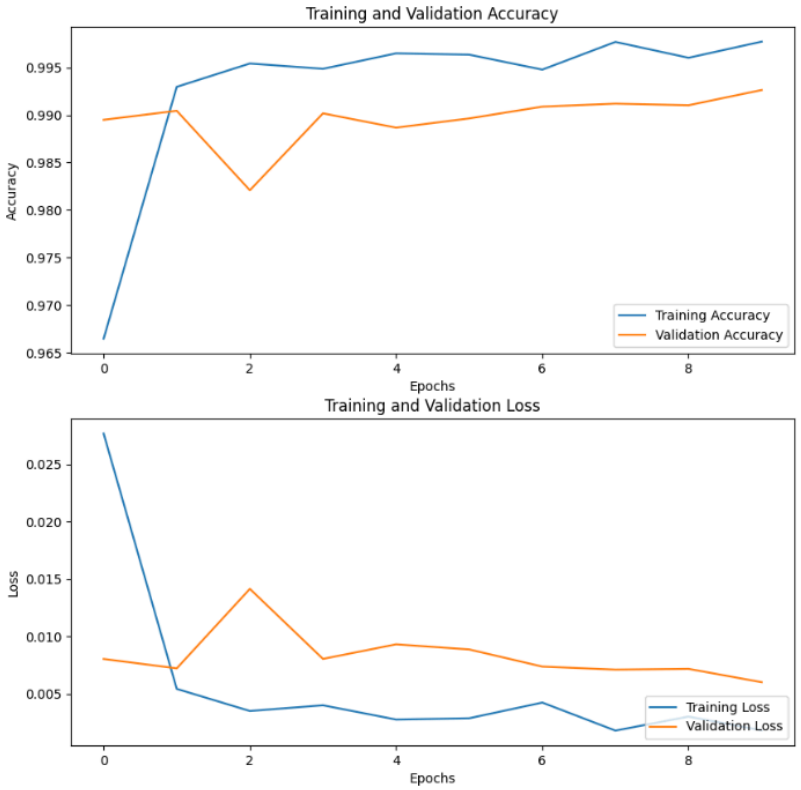
Posteriormente à escolha da CNN, o treinamento do modelo foi ampliado, isto é, foi usado o *Dataset* inteiro, usando os mesmos hiperparâmetros e computador da Tabela 1, visando aumentar sua eficácia, sendo esta fase crucial para os resultados do projeto. No entanto, mesmo com grandes quantidades de dados, a rede neural ficou enviesada, com acurácias próximas a 99% ainda nos primeiros ciclos de treinamento. Os resultados do treinamento (acurácia e taxa de erro) podem ser observados na Figura 9.

Assim, foi feita uma investigação sobre as possíveis causas desse comportamento, onde foram testadas diferentes abordagens, como:

- Geração de espectrogramas de áudios provenientes de *Datasets* como o “VCTK” (VEAUX, 2017) e o “ASVspoof 2019” (YAMAGISHI et al., 2019), que possuem, respectivamente, gravações reais e falas sintéticas na língua inglesa, sendo esses dados adicionados ao nosso *Dataset*;
- Remoção dos dados com ruído, alimentando a rede neural somente com imagens de áudios brutos;

- Uso de espectrogramas coloridos, em vez de na escala cinza.

Figura 9 - Treinamento CNN: Versão completa do *Dataset* inicial




Fonte: Imagem dos autores

Em todos os testes realizados, o modelo continuou com precisão acima do esperado. Isso levou à revisão da sua arquitetura, da forma como os espectrogramas estavam sendo gerados, e mesmo da metodologia empregada na detecção de *deepfakes*. É válido notar que o modelo teve um pior desempenho para os áudios com ruído em relação aos sem, confirmando que eles prejudicam a identificação.

Para verificar se o problema residia na arquitetura convolucional, foi decidido treinar o modelo sob as mesmas condições do treinamento primário citado anteriormente, porém utilizando o *Dataset* "*Urban Sound*" (SALAMON, 2014), que contém espectrogramas de 27 horas de áudio pertencentes a 10 classes distintas. As imagens do mesmo são coloridas e com dimensões superiores às do *Dataset* da pesquisa usado até então.

O desempenho do modelo não foi satisfatório nos primeiros *epochs*, com precisão abaixo de 1%, o que indicou que o *overfitting* ocorrido anteriormente durante o treinamento estava relacionado aos dados, e não ao modelo.

Desse modo, foram gerados espectrogramas semelhantes aos do *"Urban Sound"* - coloridos e com dimensões maiores no eixo do tempo - mais especificamente, 512px por 256px - para todas as vozes e *deepfakes* do nosso *Dataset*. Foi então realizado um treinamento visando observar o comportamento do modelo. Os hiperparâmetros usados podem ser visualizados na Tabela 2.

Tabela 2 - Hiperparâmetros e seus respectivos valores utilizados no treinamento da CNN com o Dataset oficial

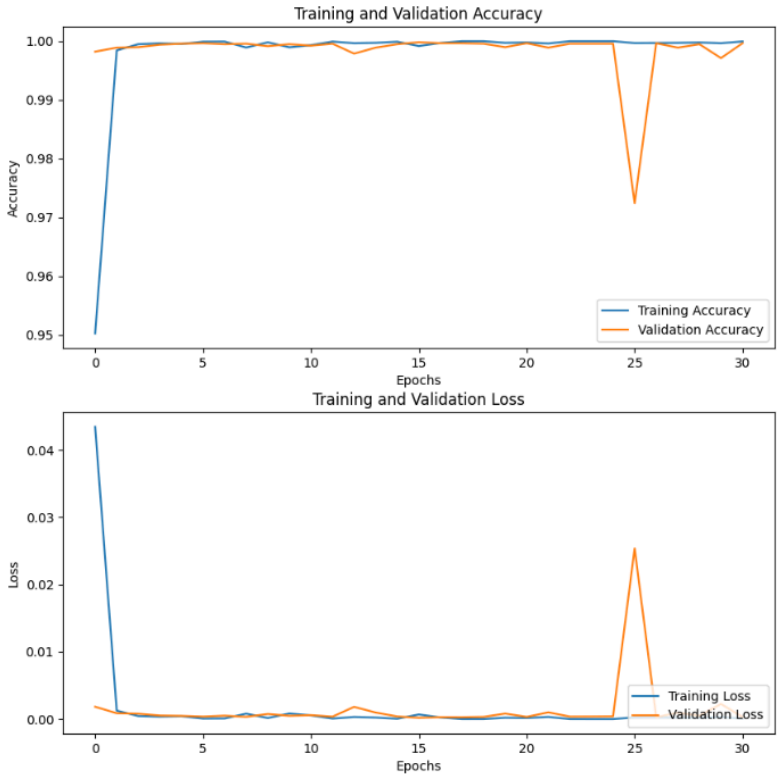
| hiperparâmetro | valor utilizado |
|---|---|
| número de *epochs* | 100 |
| tamanho do *batch* | 128 |
| taxa de aprendizagem | 0.01 |
| paciência | 15 iterações |
| função de *loss* | *Categorical Cross Entropy* |
| optimizador | *Adam* |

Fonte: Tabela dos autores

Sob essas condições, ele teve mais dificuldade em distinguir as *deepfakes* das vozes reais, com uma precisão no primeiro ciclo de treinamento de 95%, ao invés de 99% como anteriormente, o que indicou que informações estavam sendo perdidas nas antigas imagens na escala cinza e com o comprimento reduzido. O resultado desse treinamento pode ser observado na Figura 10.

Como o produto desses testes foi satisfatório, decidimos tomar essa última abordagem como

Figura 10 - Treinamento da CNN usando *Dataset* oficial sem ruídos




Fonte: Imagem dos autores

Paralelamente ao processo de treinamento do modelo, uma aplicação web foi desenvolvida. Seu objetivo é permitir aos usuários, de qualquer idade, instrução ou localidade, selecionarem arquivos de áudio que desejam classificar. Com isso, os áudios são enviados a uma API também desenvolvida.

Para chegar a esse resultado, foi iniciado um projeto com o *framework Next.js* que permite um desenvolvimento fluido de páginas web. Ademais, foram utilizados recursos essenciais como: a linguagem de programação *Typescript*, para proporcionar tipagem ao código e assim evitar que erros aconteçam; e o *Tailwind CSS*, para a estilização através de classes dos componentes das páginas.

Além disso, foram adicionadas bibliotecas, a citar: *Shadcn UI*, para a adição de componentes visuais padronizados e elegantes; *Content Layer*, para a adição de conteúdo *Markdown*; *Next Themes,* para permite a troca entre os temas claro e escuro; e *Axios*, para a comunicação com o *backend*; entre outros.

Posteriormente, foram desenvolvidos os componentes *Header* e *Footer*, para permitir que o usuário possa navegar pelas páginas, bem como para melhorar a divulgação do nome do site. Além disso, uma tabela informativa foi adicionada na página principal para comunicar algumas conquistas do projeto, como o tamanho do *Dataset*, a acurácia do modelo, as arquiteturas testadas, etc.

Após isso, foi criada a página de classificação, onde o usuário seleciona os arquivos de áudio de seu dispositivo, a partir de uma área de *input*. Ao clicar no botão “classificar”, estes são enviados para a API. É importante notar que validações foram adicionadas, para garantir que, em primeiro lugar, formatos que não sejam próprios de arquivos áudio (*wav*, *mp3*, *ogg*, *m4a*) não sejam permitidos, e, segundamente, que mais de 3 arquivos não sejam selecionados, visto que esse foi o limite definido para a classificação. Para fazer a comunicação com a API, utiliza-se a biblioteca *Axios*, que entrega informações úteis durante o upload dos arquivos, como o progresso do mesmo, a ocorrência ou não de erros, entre outras.

Em seguida, foi adicionada a página de sobre o projeto. Nela, arquivos escritos em *Markdown* são compilados para elementos da web, gerando páginas em estilo de documentação. Foram adicionadas, então, informações sobre o desenvolvimento da pesquisa, os objetivos propostos e alcançados, os resultados objetivos, etc. Para isso, foi utilizada a biblioteca *Content Layer,* que faz a conversão dos arquivos *md* e *mdx* para código *JavaScript,* que pode, então, ser renderizado na página.

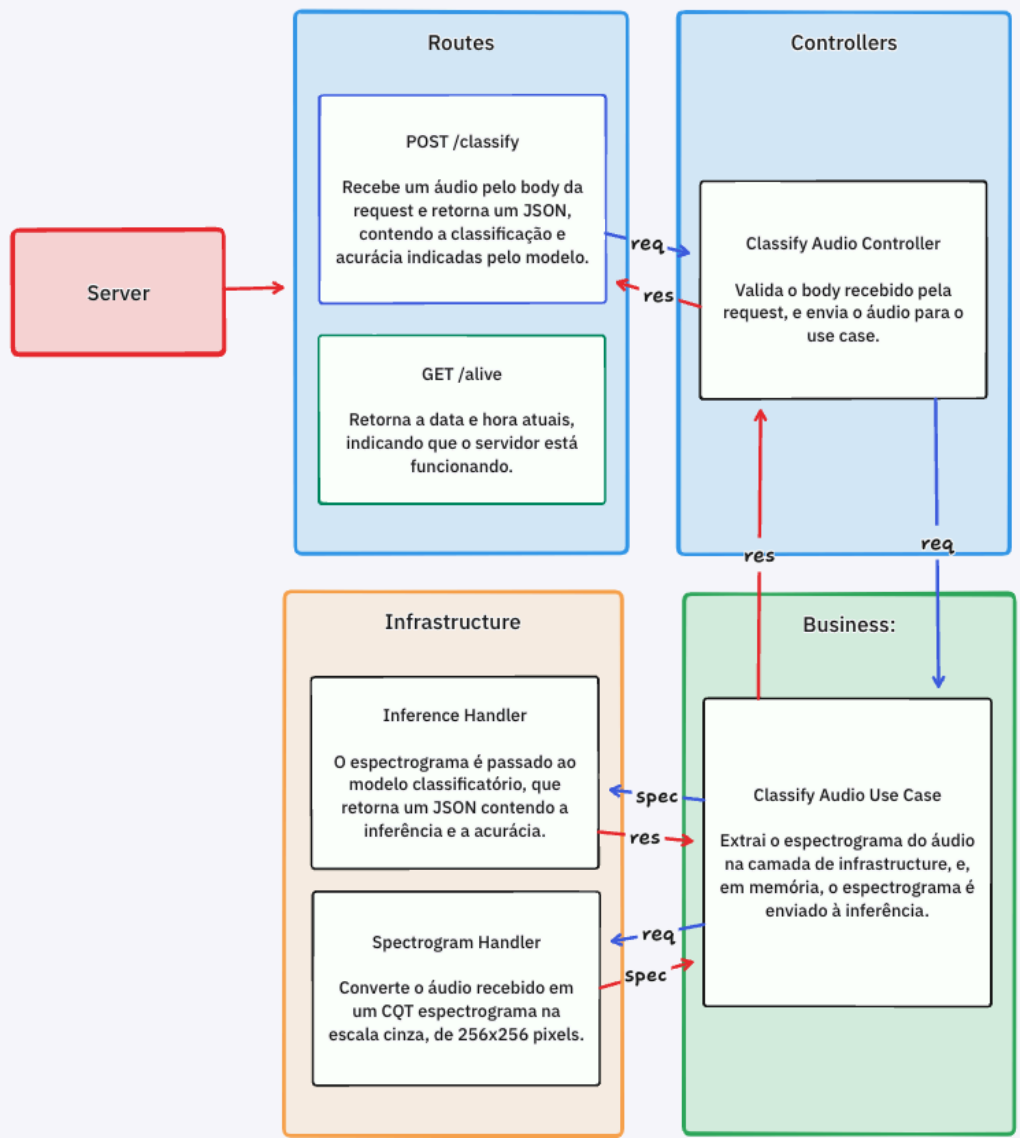
Na sequência, foi hora de adicionar, na página principal, mais elementos que pudessem dar ao usuário uma sensação de confiabilidade e credibilidade do projeto. Para esse fim, foram adicionados *Cards*, com áudios que serão classificados com o modelo final, juntamente com a acurácia e a previsão da classificação. Além disso, o usuário pode ouvir esses áudios para comparar com a veracidade dos mesmos. Não apenas isso, mas um globo terrestre foi incluído para passar a ideia de que a pesquisa visa ter alcance global, permitindo ajudar a qualquer um, em qualquer lugar.

Para concluir as páginas a serem adicionadas, foi criada a página de autores. Nela, foram adicionadas as informações, como foto, nome, descrição e contatos de cada um dos participantes do projeto, bem como dos orientadores do mesmo.

Já com relação à API, esta foi desenvolvida em *Python* utilizando o *framework Flask*, que nos permitiu criar esse elo. Para fazer a classificação, foi criado um *controller* - que é a parte do backend que recebe a requisição - para validar os parâmetros passados (garantir que há de fato um arquivo passado) e extrair o áudio dos mesmos. Este arquivo é então passado para uma lógica onde do áudio é extraído o espectrograma com o uso da biblioteca *Librosa*. Os *bytes* do espectrograma são, em memória, encaminhados para o módulo onde a classificação ocorrerá, de fato. No *inference_handler*, o modelo classificatório é carregado e o áudio é então classificado. Os resultados desse processo (a acurácia e a previsão) são então retornados para o frontend - parte visual do software e que é o objeto de interação com o usuário, neste caso, o website. A arquitetura da API pode ser observada na Figura 11.

Partiu-se, então, para a hospedagem da API no *Render,* que é uma plataforma para hospedagem gratuita de serviços. Entretanto, após ter sido configurada e hospedada, notou-se o seguinte problema: após 15 minutos sem receber requisições, o servidor no qual a API está hospedada entra em inatividade. Quando isso acontece, aquele demora em média de 1 minuto para voltar à atividade, o que implica em um grande aumento do tempo de resposta, o que pode piorar a experiência do usuário; ao voltar à atividade, as respostas chegam rapidamente. Para contornar o empecilho, decidiu-se criar um script *Keep Alive*, que é um programa escrito em Python e que, a cada 14 minutos, faz uma requisição para a API enviando um áudio de exemplo. Assim, é garantido que, enquanto o programa estiver rodando, o servidor não entrará em inatividade. Com isso, bastou compilar o mesmo e adicioná-lo a uma tarefa automática no computador, para que, quando este ligar, o script já comece a rodar em segundo plano no dispositivo.

Figura 11 - Estrutura da *API* utilizada para conexão com o *website*




Fonte: Imagem dos autores

Fica claro, portanto, que se trata de uma API simples, quando comparada com as utilizadas em projetos mais complexos, visto que tem apenas uma rota, de onde chega o áudio enviado pelo usuário e de onde sairá a resposta da classificação para o mesmo. Entretanto, é desta simplicidade que o projeto precisa, já que seu foco é no desenvolvimento de uma inteligência artificial eficaz e robusta que possa entregar resultados satisfatórios ao usuário final.

Cabe destacar que o modelo presente na API não é o mais recente desenvolvido, mas sim a versão inicial treinada com espectrogramas em preto e branco de dimensões 256px por 256px. Isso pois a plataforma de hospedagem utilizada possui um limite gratuito de 500 MB de memória, e o modelo atual necessita de um valor maior. No entanto, a versão anterior também possui uma eficácia elevada, o que acreditamos que não impactará significativamente no produto final.

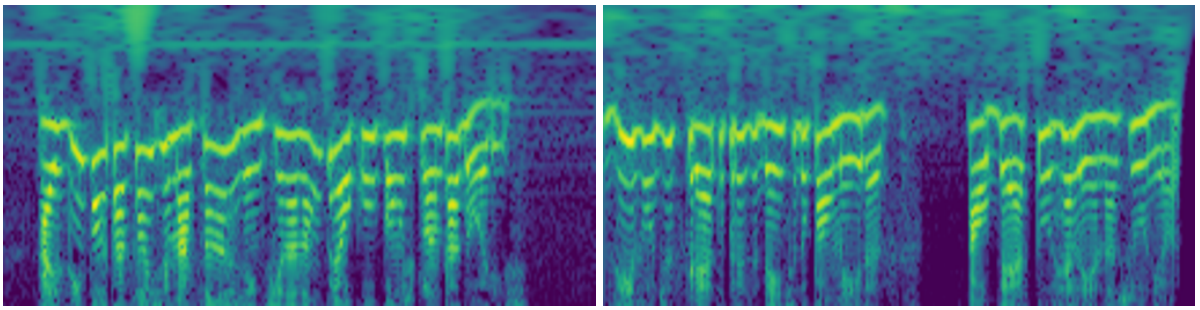
Por fim, a documentação da API pode ser consumida no seguinte link: https://github.com/Unfake-Official/server, enquanto a documentação do website está presente nessa página: https://github.com/Unfake-Official/web.

## RESULTADOS

Neste trabalho, foi desenvolvido um *Dataset* contendo espectrogramas de deepfakes e gravações de falas reais, sendo nomeado de “Portufake”. Ele contém espectrogramas CQT com dimensões de 256px por 512px de ambos áudios reais e falsos, sendo cada imagem associada à sua categoria: *real* (real) ou *fake* (falso). Também foram adicionados ruídos em 16.7% das imagens, para cada voz, respeitando a metodologia citada anteriormente.

Exemplos de imagens dos espectrogramas podem ser visualizados na Figura 12, onde, à esquerda, observa-se uma representação de fala real e, à direita, de fala sintética. No total, o “Portufake” possui 183.878 espectrogramas, totalizando 33,4 GB. A relação entre locutores masculinos e femininos, bem como de alterações nos áudios do *Dataset*, está presente na Tabela 3.

Figura 12 - Exemplo de versão final dos espectrogramas reais e falsos



Fonte: Imagem dos autores

Tabela 3 - Número e relação de imagens do “Portufake” em relação às alterações adicionadas e ao gênero dos locutores

| Alterações/Gênero | Masculino (42 vozes) | Feminino (49 vozes) | Total (91 vozes) |
|---|---|---|---|
| Sem alterações | 75.996 | 79.626 | 157.622 |
| Com ruídos | 12.664 | 13.592 | 26.256 |
| Total | 88.660 | 95.218 | 183.878 |

Fonte: Tabela dos autores

Ademais, foi proposta uma abordagem que utiliza visão computacional para detectar *deepfakes* de áudio na língua portuguesa, sendo treinada e validada uma rede neural convolucional (CNN) para esse fim, a qual foi nomeada de “Unfake”. O modelo atual conta com 121.924.068 parâmetros, adquirindo, após o último treinamento, 465,10 MB de tamanho, e precisão de 99% para o conjunto de dados desenvolvido.

O desempenho do modelo para diferentes conjuntos de testes pode ser visualizado na Tabela 4, onde observa-se a precisão e erro quadrático médio do mesmo. Foram realizados três tipos de testes, onde o modelo recebeu como *input*:

- Todas as imagens do “Portufake”;
- Todas as imagens do “Portufake” sem ruídos;
- Espectrogramas CQT coloridos de 256px por 512px criados a partir dos *datasets* “ASVspoof 2019” e “VCTK”. No caso do “ASVspoof 2019”, foram utilizados todos os áudios de *evaluation* da categoria *Logical Access (LA)* - que contém *deepfakes* criadas com uso de ferramentas *Text-to-Speech e Voice Conversion* - presente no *dataset*.

Tabela 4: Precisão e erro do “Unfake” para diferentes casos de teste

| | Precisão (*Categorical Accuracy*) | Erro *(Mean Squared Error*) |
|---|---|---|
| *Dataset* completo | 99,843% | 0,122% |
| *Dataset* sem ruídos | 99,833% | 0,125% |
| VCTK + ASVspoof 2019 | 38,844% | 60,383% |

Fonte: Tabela dos autores

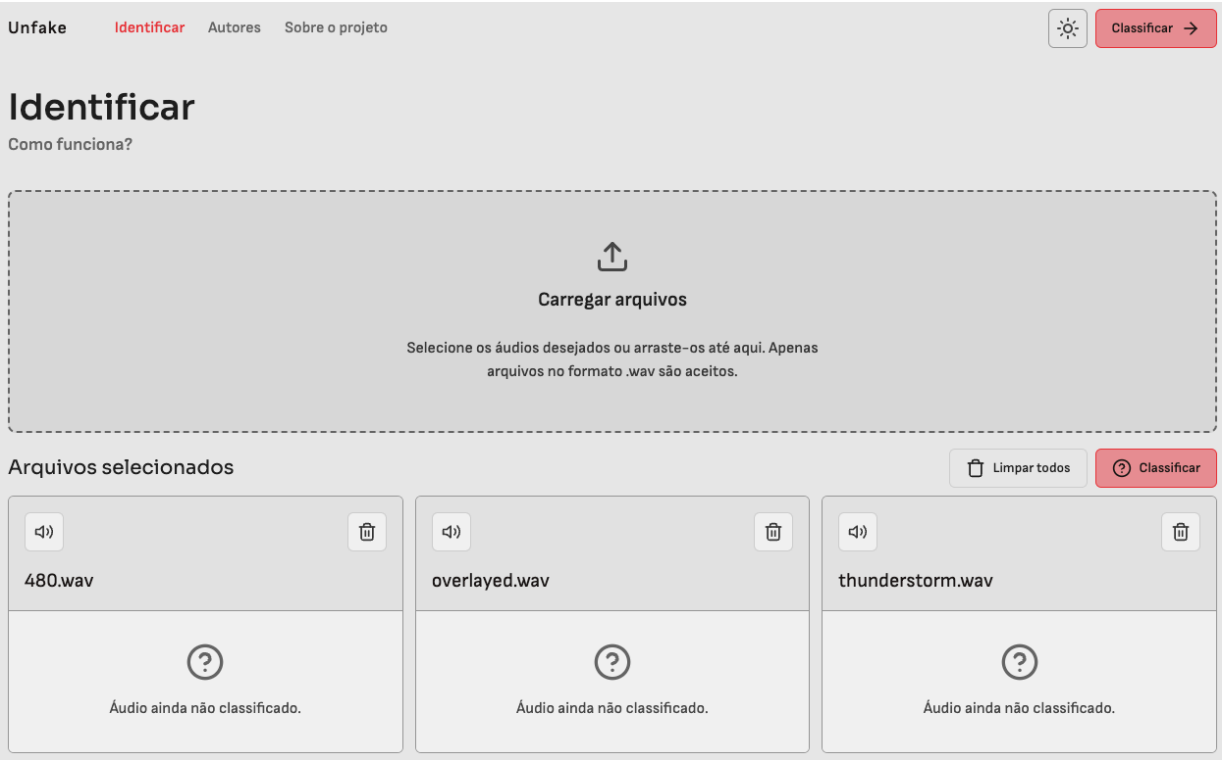
Fazendo-se uma análise quantitativa dos resultados, é possível observar que o modelo “Unfake” ficou enviesado ao conjunto de dados usado no seu treinamento, o que o levou a ter uma acurácia extremamente alta. Isto se deve ao fato de ter-se utilizado somente o modelo “XTTS” para a geração das *deepfakes*, de modo que nossa rede está limitada a um tipo de *deepfake* específico.

Além disso, percebe-se um desempenho limitado para dados não vistos, já que, quando foi submetida ao conjunto de dados “VCTK + ASVSpoof 2019”, que contém espectrogramas de gravações e *deepfakes* em inglês, a rede neural teve precisão mais de 3 vezes menor, e taxa de erro cerca de 1200 vezes maior. Isso pode ser explicado tanto pelo fato dos modelos utilizados na geração das *deepfakes* serem distintos quanto da língua usada no teste ser diferente da rede neural.

Ademais, nota-se que o modelo teve desempenho ligeiramente maior quando foi submetido ao *dataset* com ruídos, sugerindo que essas alterações são benéficas para o modelo, aumentando a sua *perfomance* quando presentes.

Por fim, foi criada uma API e um website que permite aos usuários utilizar a versão inicial da rede neural, democratizando a detecção de *deepfakes* entre a população brasileira. O site, funcional e público, está disponível no link: https://unfake-web.vercel.app. A página de identificar *deepfakes* demonstrando a principal funcionalidade do app pode ser observada na Figura 13.

Figura 13: Página para identificar *deepfakes*




Fonte: Imagem dos autores

## CONCLUSÕES

Os objetivos do presente trabalho foram parcialmente alcançados, visto que foi desenvolvido: um *dataset* de *deepfakes* em português, pioneiro na língua; um modelo CNN que permite identificar *deepfakes* semelhantes às do *Dataset*, ainda que esteja limitado; e um *website* em conjunto com uma API que permite o uso, por usuários leigos, dessa ferramenta. Vale lembrar que todos os links para os repositórios do projeto estão presentes no Anexo A.

A respeito da metodologia empregada, é possível concluir que ela pode, de fato, ser utilizada na classificação de áudios e identificação de *deepfakes*, já que os resultados demonstram uma melhora do modelo em classificar os dados ao longo do tempo. Já em relação aos áudios, conclui-se que a introdução de ruídos nos mesmos é benéfica para o modelo, dado que ele tem mais facilidade em classificar conteúdos com essas alterações. Já considerando os espectrogramas, é possível perceber que os que possuem escala colorida e dimensões maiores no eixo do tempo levam a resultados mais robustos, em comparação com os que têm dimensões menores e são criados na escala cinza.

Outrossim, conclui-se que que o projeto desenvolvido pode auxiliar a mitigar os efeitos nocivos provocados pelas *deepfakes* de áudio nos setores social, político e econômico, à medida em que permite a verificação, ainda que limitada, de conteúdos encontrados nas redes. Além disso, a produção de um *dataset* de *deepfakes* em português pode incentivar novas pesquisas na área, tendo em mente que elas podem utilizar os dados produzidos para fins de classificação, análise estatística, entre outros.

Por fim, os passos futuros para o aprimoramento da pesquisa incluem:

- A continuidade do treinamento do modelo desenvolvido com espectrogramas provenientes de áudios com ruídos, com sons ambientes e com sons ambientes adicionados de ruídos;
- A descoberta de uma plataforma para hospedagem da API que permita utilizar versões do modelo que exigem mais memória e processamento;

- A utilização de mais de uma abordagem *Text-To-Speech* ou *Voice Conversion* para a geração das *deepfakes* em português, além do modelo “XTTS”, garantindo que a solução seja capaz de generalizar para mais de um padrão de áudio sintético;
- O uso de dados gerados com *deepfake* através de plataformas como o Youtube ou Instagram, desde que devidamente e confiavelmente categorizados, caso seja possível, permitindo a expansão do *Dataset* de treinamento sem a necessidade de geração de *deepfakes*;
- E a testagem de outras metodologias para a identificação de áudios falsos, incluindo aquelas não baseadas em visão computacional, como comparação estatística, uso de algoritmos de processamento de sinais, e modelos de aprendizagem profunda que operam em áudios brutos.

## REFERÊNCIAS BIBLIOGRÁFICAS


ALMUTAIRI, Zaynab; ELGIBREEN, Hebah. A review of modern audio deepfake detection methods: Challenges and future directions. **Algorithms**, v. 15, n. 5, p. 155, 2022. Disponível em: <https://www.mdpi.com/1999-4893/15/5/155>. Acesso em: 9 mar. 2024.

ARDILA, Rosana et al. Common voice: A massively-multilingual speech corpus. **arXiv preprint arXiv:1912.06670**, 2019. Disponível em: <https://arxiv.org/pdf/1912.06670>. Acesso em: 10 mar. 2024.

AUDIO Corpora - Bases de áudio. **Gitlab**. Disponível em: <https://gitlab.com/fb-audio-corpora>. Acesso em: 7 mar. 2024.

BRITT, Kaeli. How are deepfakes dangerous?. **Nevada Today**, 2023. Disponível em: <https://www.unr.edu/nevada-today/news/2023/atp-deepfakes>. Acesso em: 15 jun. 2024.

CASANOVA, Edresson et al. XTTS: a Massively Multilingual Zero-Shot Text-to-Speech Model. **arXiv e-prints**, p. arXiv: 2406.04904, 2024. Disponível em: <https://arxiv.org/pdf/2406.04904>. Acesso em: 11 jun. 2024.

DOSOVITSKIY, Alexey et al. An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020. Disponível em: <https://arxiv.org/pdf/2010.11929>. Acesso em: 20 jul. 2024.

DISTRIBUIÇÃO normal. In: **WIKIPÉDIA: a enciclopédia livre**. Disponível em: <https://pt.wikipedia.org/wiki/Distribuição_normal>. Acesso em: 30 abr. 2024.

PICZAK, Karol J. ESC: Dataset for environmental sound classification. In: **Proceedings of the 23rd ACM international conference on Multimedia**. 2015. p. 1015-1018. Disponível em: <https://www.karolpiczak.com/papers/Piczak2015-ESC-Dataset.pdf>. Acesso em: 23 abr. 2024.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological**

**Cybernetics**, v. 36, n. 4, p. 193–202, abr. 1980. Disponível em: <https://www.cs.princeton.edu/courses/archive/spr08/cos598B/Readings/Fukushima1980.pdf>. Acesso em: 25 abr. 2024

GÖLGE, Eren et al. Coqui TTS (Version 1.4) [Computer software]. **Zenodo**, 2022. Disponível em: <https://doi.org/10.5281/zenodo.6334862>. Acesso em: 15 mai. 2024.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep Learning. **MIT Press**, 2016. Disponível em: <https://www.deeplearningbook.org/>. Acesso em: 15 abr. 2024.

HAQUE, Ayaan. Artificial Data for Image Classification. **Towards Data Science**, 2020. Disponível em: <https://towardsdatascience.com/artificial-data-for-image-classification-5b2ede40640f>. Acesso em: 25 mai. 2024.

HAQUE, Ayaan. EC-GAN: Low-Sample Classification using Semi-Supervised Algorithms and GANs. **arXiv preprint arXiv:2012.15864**, 2021. Disponível em: <https://arxiv.org/pdf/2012.15864>. Acesso em: 25 mai. 2024.

HONG, T. J. Uncovering the Real Voice: How to Detect and Verify Audio Deepfakes. **Medium**, 2023. Disponível em: <https://medium.com/htx-s-s-coe/uncovering-the-real-voice-how-to-detect-and-verify-audio-deepfakes-42e480d3f431>. Acesso em: 5 mar. 2024.

IOFFE, Sergey. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015. Disponível em: <https://arxiv.org/pdf/1502.03167>. Acesso em: 2 jun. 2024.

KAWA, Piotr; PLATA, Marcin; SYGA, Piotr. Attack Agnostic Dataset: Towards Generalization and Stabilization of Audio DeepFake Detection. **arXiv preprint arXiv:2206.13979**, 2022. Disponível em: <https://arxiv.org/pdf/2206.13979v2>. Acesso em: 20 mar. 2024.

KAWA, Piotr; PLATA, Marcin; SYGA, Piotr. Defense Against Adversarial Attacks on Audio DeepFake Detection. **arXiv preprint arXiv:2212.14597**, 2022. Disponível em: <https://arxiv.org/pdf/2212.14597v2>. Acesso em: 21 mar. 2024.

KEVIN, Chris. Feature Maps. **Medium**, 2018. Disponível em: <https://medium.com/@chriskevin_80184/feature-maps-ee8e11a71f9e>. Acesso em: 30 abr. 2024.

KIM, J. et al. Glow-TTS: A generative flow for text-to-speech via monotonic alignment search. **arXiv preprint arXiv:2005.11129**, 2020. Disponível em: <https://arxiv.org/pdf/2005.11129>. Acesso em: 4 abr. 2024.

KHANJANI, Zahra; WATSON, Gabrielle; JANEJA, Vandana P. Audio deepfakes: A survey. **Frontiers in Big Data**, v. 5, p. 1001063, 2023. Disponível em: <https://www.frontiersin.org/articles/10.3389/fdata.2022.1001063/full>. Acesso em: 9 mar. 2024.

KONG, J.; KIM, J.; BAE, J. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. **arXiv preprint arXiv:2010.05646**, 2020. Disponível em: <https://arxiv.org/pdf/2010.05646>. Acesso em: 4 mar. 2024.

LEAL, R. DOS S. Datasets de Áudio em Português. **Medium**, 2021. Disponível em: <https://medium.com/@renatoleal/datasets-de-%C3%A1udio-em-portugu%C3%AAs-b25316ec316a>. Acesso em: 6 mar. 2024.

LECUN, Yann. Generalization and network design strategies. **Connectionism in perspective**, v. 19, n. 143-155, p. 18, 1989. Disponível em: <http://yann.lecun.com/exdb/publis/pdf/lecun-89.pdf>. Acesso em: 25 abr. 2024.

LEFFER, L. AI Audio Deepfakes Are Quickly Outpacing Detection. **Scientific American**, 2024 Disponível em: <https://www.scientificamerican.com/article/ai-audio-deepfakes-are-quickly-outpacing-detection/>. Acesso em: 5 mar. 2024.

LI, Xiang et al. Understanding the disharmony between dropout and batch normalization by variance shift. **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**, 2019. p. 2682-2690. Disponível em:

<https://openaccess.thecvf.com/content_CVPR_2019/papers/Li_Understanding_the_Disharmony_Between_Dropout_and_Batch_Normalization_by_Variance_CVPR_2019_paper.pdf>. Acesso em: 19 mar. 2024.

MARTINS, Américo. Eleições nos EUA: uso de deepfake e IA revela problema que pode se repetir no Brasil. **CNN Brasil**, 2024. Disponível em: <https://www.cnnbrasil.com.br/internacional/eleicoes-nos-eua-uso-de-deepfake-e-ia-revela-problema-que-pode-se-repetir-no-brasil/>. Acesso em: 10 jun. 2024.

MASOOD, Momina et al. MASOOD, Momina et al. Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward. **Applied intelligence**, v. 53, n. 4, p. 3974-4026, 2023. Disponível em: <https://arxiv.org/pdf/2103.00484>. Acesso em: 10 mar. 2024.

MUBARAK, Rami et al. A Survey on the Detection and Impacts of Deepfakes in Visual, Audio, and Textual Formats. **IEEE Access**, 2023. Disponível em: <https://ieeexplore.ieee.org/document/10365143?denied=>. Acesso em: 9 mar. 2024.

MÜLLER, Nicolas M. et al. Does audio deepfake detection generalize?. **arXiv preprint arXiv:2203.16263**, 2022. Disponível em: <https://arxiv.org/abs/2203.16263>. Acesso em: 10 mar. 2024.

OLIVEIRA, Rafael Santana. Desenvolvimento de recursos e ferramentas para reconhecimento de voz em português brasileiro para Desktop e Sistemas Embarcados. **Universidade Federal do Pará. Belém**, 2012. Disponível em: <https://ppgcc.propesp.ufpa.br/Dissertações_2012/Rafael%20Santana%20Oliveira_Dissertação.pdf>. Acesso em: 3 mar. 2024.

PANDEY, Swarnima. How to choose the size of the convolution filter or Kernel size for CNN?. **Medium**, 2020. Disponível em: <https://medium.com/analytics-vidhya/how-to-choose-the-size-of-the-convolution-filter-or-kernel-size-for-cnn-86a55a1e2d15>. Acesso em: 30 abr. 2024.

PRADO, Magaly Pereira do. Deepfake de áudio: manipulação simula voz real para retratar alguém dizendo algo que não disse. **TECCOGS – Revista Digital de Tecnologias Cognitivas**, 2021, p. 45-68. Disponível em:

<https://revistas.pucsp.br/index.php/teccogs/article/view/55977/37926>. Acesso em: Acesso em: 2 mar. 2024.

RALLABANDI, Srikari. Activation functions: ReLU vs. Leaky ReLU. **Medium**, 2023. Disponível em: <https://medium.com/@sreeku.ralla/activation-functions-relu-vs-leaky-relu-b8272dc0b1be>. Acesso em: 25 abr. 2024.

SALAMON, Justin; JACOBY, Christopher; BELLO, Juan Pablo. A dataset and taxonomy for urban sound research. In: **Proceedings of the 22nd ACM international conference on Multimedia**. 2014. p. 1041-1044. Disponível em: <https://www.justinsalamon.com/uploads/4/3/9/4/4394963/salamon_urbansound_acmmm14.pdf>. Acesso em: 10 set. 2024.

SCHILLING, Fabian. The Effect of Batch Normalization on Deep Convolutional Neural Networks. **KTH Royal Institute of Technology**, 2016. Disponível em: <https://www.diva-portal.org/smash/get/diva2:955562/FULLTEXT01.pdf>. Acesso em: 18 mai. 2024.

SCHÖRKHUBER, Christian; KLAPURI, Anssi. Constant-Q transform toolbox for music processing. In: **7th sound and music computing conference, Barcelona, Spain.** SMC, 2010. p. 3-64. Disponível em: <https://www.researchgate.net/profile/Christian-Schoerkhuber/publication/228523955_Constant-Q_transform_toolbox_for_music_processing/links/55126aa60cf20bfdad513a3f/Constant-Q-transform-toolbox-for-music-processing.pdf>. Acesso em: 14 mai. 2024.

SHEN, J. et al. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. **arXiv preprint arXiv:1712.05884**, 2017. Disponível em: <https://arxiv.org/pdf/1712.05884>. Acesso em: 5 mar. 2024.

SOUZA, Thiago Campolina de. Como inteligência artificial, deepfakes e agências de checagem atuam na arena da desinformação. **Jornal da USP**, 04 nov. 2022. Disponível em: <https://jornal.usp.br/ciencias/como-inteligencia-artificial-deepfakes-e-agencias-de-checagem-atuam-na-arena-da-desinformacao/>. Acesso em: 2 mar. 2024.

SRIVASTAVA, Nitish. Improving Neural Networks with Dropout. **University of Toronto, 2013**. Disponível em: <https://www.cs.toronto.edu/~nitish/msc_thesis.pdf>. Acesso em: 21 mai. 2024.

STUPP, Catherine. Fraudsters used AI to mimic CEO's voice in unusual cybercrime case. **The Wall Street Journal**, v. 30, n. 08, 2019. Disponível em: <https://fully-human.org/wp-content/uploads/2019/10/Stupp_Fraudsters-Used-AI-to-Mimic-CEOs-Voice-in-Unusual-Cybercrime-Case.pdf>. Acesso em: 9 mar. 2024.

THAMBAWITA, Vajira et al. Impact of Image Resolution on Deep Learning Performance in Endoscopy Image Classification: An Experimental Study Using a Large Dataset of Endoscopic Images. **Diagnostics**, 2021. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246/>. Acesso em: 20 abr. 2024.

VAN DEN OORD, A. et al. WaveNet: A generative model for raw audio. **arXiv preprint arXiv:1609.03499**, 2016. Disponível em: <https://arxiv.org/pdf/1609.03499>. Acesso em: 4 mar. 2024.

VEAUX, Christophe et al. CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit. University of Edinburgh. **The Centre for Speech Technology Research (CSTR)**, v. 6, p. 15, 2017. Disponível em: <https://datashare.ed.ac.uk/handle/10283/2651>. Acesso em: 16 fev. 2024.

WANG, Pin; FAN, En; WANG, Peng. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. **Pattern Recognition Letters**, v. 141, p. 61-67, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865520302981>. Acesso em: 17 mar. 2024.

YAMAGISHI, Junichi et al. Asvspoof 2019: The 3rd automatic speaker verification spoofing and countermeasures challenge database, 2019. Disponível em: <https://erepo.uef.fi/handle/123456789/7718>. Acesso em: 16 fev. 2024.

## ANEXOS

### ANEXO A - Tabela de repositórios do projeto

| Nome | Descrição | Link |
|---|---|---|
| library | Contém os programas usados na geração de *deepfakes* e na elaboração dos *Datasets*. | https://github.com/Unfake-Official/library |
| classifier | Contém os programas para treinamento e inferência de diferentes modelos testados (CNN, GAN e Vision-Transformer). | https://github.com/Unfake-Official/classifier |
| server | Contém os arquivos da API do projeto. | https://github.com/Unfake-Official/server |
| web | Contém os arquivos do website do projeto. | https://github.com/Unfake-Official/web |
| fake_voices | Contém o *Dataset* com *deepfakes* criadas usando o "XTTS". | https://huggingface.co/datasets/unfake/fake_voices |
| portufake | Contém três *datasets*:<br>● O "Portufake";<br>● A versão inicial do "Portufake" usada no treinamento da versão inicial do modelo "Unfake";<br>● E o *Dataset* com espectrogramas do "VCTK" e do "ASVspoof 2019". | https://huggingface.co/datasets/unfake/portufake |
| unfake | Contém dois modelos:<br>● O "Unfake";<br>● E a versão inicial do "Unfake", presente na API do website. | https://huggingface.co/unfake/unfake |

## ANEXO B - Glossário

***Batch***: Define o número de amostras, ou linha de dados, a percorrer antes de atualizar os parâmetros internos de uma rede neural em seu ciclo de treinamento.

**Camada (rede neural):** Conjunto de neurônios ou unidades de processamento que aplicam uma transformação nos dados de entrada e os passam para a próxima camada de uma rede neural.

***Epoch***: Define o número de vezes que um algoritmo de aprendizado percorrerá seu conjunto de treinamento inteiro.

**Função de ativação:** É uma função que calcula a saída de um neurônio em uma rede neural com base em suas entradas individuais e seus pesos.

**Função de *loss*:** É uma função que avalia a *performance* de uma rede neural, resultando em um custo ou *loss* mais alto para previsões ruins, e mais baixo para previsões boas.

**Hiperparâmetro:** Parâmetro que pode ser definido para configurar o processo de aprendizado de uma rede neural.

**Optimizador:** Função ou algoritmo que ajusta os atributos e pesos de uma rede neural, reduzindo a *loss* e auxiliando na melhora da precisão do modelo.

**Paciência:** Define o número de *epochs* a serem executados em um ciclo de treinamento depois que a *loss* parou de diminuir.

**Taxa de aprendizagem:** Define a velocidade para qual o algoritmo de otimização converge para os pesos ideais dos neurônios de uma rede neural.