



Celso Gabriel Vieira Ribeiro Lopes

Métodos de Detecção de Imagens *DeepFake* baseadas em Modelos Generativos

São José dos Campos, SP

Celso Gabriel Vieira Ribeiro Lopes

Métodos de Detecção de Imagens *DeepFake* baseadas em Modelos Generativos

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência de Tecnologia

Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Fabio Augusto Faria

São José dos Campos, SP

Fevereiro de 2022

Celso Gabriel Vieira Ribeiro Lopes

Métodos de Detecção de Imagens *DeepFake* baseadas em Modelos Generativos

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado em 10 de Fevereiro de 2022:

Prof. Dr. Fabio Augusto Faria
Orientador

Prof. Dr. Sérgio Ronaldo Barros dos Santos
Convidado 1

Prof. Dr. Luis Augusto Martins Pereira
Convidado 2

São José dos Campos, SP
Fevereiro de 2022

*Este trabalho é dedicado as pessoas que me apoiaram durante minha trajetória acadêmica.
Em especial minha mãe Maria Alice Vieira Ribeiro Lopes e meu pai Celso Ribeiro Lopes.*

Agradecimentos

Agradeço primeiramente a Deus por me dar força durante essa trajetória da faculdade. Quero agradecer ao meu professor e orientador Dr. Fabio Augusto Faria e ao professor Dr. Raoni Florentino da Silva Teixeira, sem ele não conseguiria realizar esse trabalho. Ele me guiou em todos os momentos que fiquei perdido e sempre me mostrou para qual lado seguir. Quero agradecer aos meus pais que sempre estiveram ao meu lado me apoiando e me dando suporte.

*“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)*

Resumo

A manipulação de imagens e vídeos não é uma tarefa recente na literatura, já que essa tarefa era executada pela utilização de ferramentas de edição, como Photoshop e GIMP, onde era facilmente possível identificar a olho nu se uma imagem era real ou não. Entretanto com o avanço da área de inteligência artificial e mais especificamente na aprendizagem profunda possibilitou o surgimento do chamado *DeepFake*. O termo *DeepFake* se refere a geração de conteúdos multimídias muito realistas onde as pessoas não conseguem diferenciar entre real e falso. Essa técnica pode ser aplicada a diversas situações como no entretenimento, na educação e em jogos. Entretanto se for utilizada por pessoas maliciosas, pode acabar acarretando em práticas antiéticas. Infelizmente o *DeepFake* ficou muito conhecido devido as más práticas realizadas com sua utilização. O primeiro surgimento da técnica ocorreu em 2017 no *Reddit*, em uma publicação com vídeos pornográficos de celebridades. Após esse caso, começou a surgir diversas *DeepFake* na internet. Tendo em vista que ela pode ser utilizada para o mal, se iniciou um movimento mundial de combate de tais práticas com a proposição de novas técnicas automatizadas de detecção das imagens e vídeos falsos. Neste trabalho foi realizado um estudo de diferentes abordagens de detecção de imagens *DeepFake*, bem como também, a implementação de duas diferentes abordagens para a tarefa alvo. Na primeira técnica utiliza de extração dos ruídos residuais por meio de técnicas PRNU, os quais serão utilizados posteriormente no treinamento de CNNs. Já o segundo método baseia-se em medidas de qualidade de imagem (Correlação, SSIM e MSE) para realizar a detecção de imagens *DeepFake*. Nos resultados realizados foi possível mostrar que a utilização de imagens originais carregam mais informações relevantes para a tarefa alvo que as imagens resultantes da extração PRNU.

Palavras-chaves: Aprendizagem profunda; DeepFake; Detecção de Deepfake; PRNU; Rede Neural Convolucional; Medidas de Qualidade da Imagem.

Abstract

The doctoring of images and videos is not a recent task in literature, since it has been performed by image manipulation tools such as Photoshop and GIMP, where it was possible to easily identify if an image was real or not with the naked eye. However, the advancements in the field of artificial intelligence and more specifically deep learning have made the emergence of the so-called DeepFake possible. DeepFake refers to the generation of realistic multimedia content which is hard for people to discern from real content. This technique can be applied to a diverse range of situations such as those in entertainment, education and games. However if used by malicious people, it can result in unethical practices. Unfortunately DeepFake became well-known due to its misuse. The first use of the technique was to create fake pornographic videos of celebrities that were published on Reddit in 2017. After this occurrence, many DeepFakes started to show up on the internet. Knowing that it can be used in negative ways, a worldwide movement aimed to stop such practices was started with the proposal of new automated techniques for detecting fake images and videos. In this work, different approaches for DeepFake image detection were studied, two of which were implemented. The first of these approaches makes use of residual noise extractions through PRNU techniques, which are then used to train CNNs. As for the second approach, it is based on image quality measures (Correlation, SSIM and MSE) to detect DeepFake images. The results show that the use of the original images as input is advantageous compared to the use of the results of PRNU extraction, since they carry more relevant information to the task at hand.

Key-words: Deep Learning; DeepFake; DeepFake Detection; PRNU; Convolutional Neural Network; Image Quality Measurements.

Listas de ilustrações

Figura 1 – Exemplo de imagem PRNU.	30
Figura 2 – Comparação dos métodos SSIM e MSE para imagens de Einstein alteradas com diferentes tipos de distorções.	31
Figura 3 – Hierarquia do Aprendizado.	33
Figura 4 – Neurônio Biológico	34
Figura 5 – <i>Perceptron</i> .	35
Figura 6 – Exemplo de camadas convolucionais	37
Figura 7 – Exemplo de convolução	38
Figura 8 – Exemplo de aplicação do <i>max-pooling</i> e average pooling de tamanho 2×2 com passo 2 sobre uma mapa de características	38
Figura 9 – Exemplo de GAN	39
Figura 10 – Exemplo de imagem real e imagem criada pela StyleGAN.	40
Figura 11 – Técnicas de estimativas baseadas na idéia de amostragem	40
Figura 12 – Passos seguidos pelo primeiro colocado na geração de <i>DeepFake</i> .	45
Figura 13 – Passos seguidos pelo segundo colocado na geração de emphDeepFake.	46
Figura 14 – Estrutura do detector de <i>DeepFake</i> do primeiro colocado.	47
Figura 15 – Estrutura do detector de <i>DeepFake</i> do segundo colocado.	48
Figura 16 – Diagrama representando os passos a serem seguidos para a detecção de <i>DeepFakes</i> utilizando CNN.	50
Figura 17 – Exemplo de recorte realizado na face.	51
Figura 18 – Diagrama representando os passos a serem seguidos para a detecção de <i>DeepFakes</i> utilizando métodos de medida de qualidade da imagem.	53
Figura 19 – Recortes realizados na face.	56
Figura 20 – Exemplo de datasets utilizados.	57
Figura 21 – Exemplo de extração de PRNU.	57
Figura 22 – Exemplo das imagens com os 4 alphas utilizados.	59
Figura 23 – Mapa de calor da correlação entre as classes.	62
Figura 24 – TSNE das imagens PRNU com suas respectivas classes.	63
Figura 25 – Aproximação do TSNE das imagens PRNU.	63

Listas de tabelas

Tabela 1 – Matriz de confusão de um classificador	41
Tabela 2 – Matriz de confusão para classificação com duas classes	41
Tabela 3 – Exemplo de acurácia top-3	42
Tabela 4 – Resultados dos <i>datasets</i> 1 e 2 sem a utilização de PRNU.	58
Tabela 5 – Resultados do <i>dataset</i> 2 utilização PRNU e resize.	59
Tabela 6 – Resultados do <i>dataset</i> 2 utilização apenas PRNU.	59
Tabela 7 – Acurácia top- <i>N</i> dos métodos Correlação, SSIM e MSE utilizando PRNU. .	60

Lista de abreviaturas e siglas

IA	Inteligência Artificial
CNN	Convolutional Neural Network (Rede Neural Convolucional)
RNA	Rede Neural Artificial
GAN	Generative Adversarial Network
AM	Aprendizado de máquina
VP	Verdadeiro Positivo
VN	Verdadeiro Negativo
FP	Falso Positivo
FN	Falso Negativo
max	Valor máximo entre dois números
TWC	Transformada de Wavelet Contínua
TWD	Transformada de Wavelet Discreta
PRNU	Photo Response Non-Uniformity
MSE	Mean Squared Error
SSIM	Structural Similarity Index
DFDC	The DeepFake Detection Challenge
DFGC	Deepfake Game Competition

Listas de símbolos

σ	Letra grega Sigma
.	Multiplicação
e	Número de Euler
μ	Letra grega Mi
π	Valor constante Pi
χ	Letra grega Chi
ω	Letra grega Omega
Ψ	Letra grega Psi
\in	Pertence
\mathbb{R}	Conjunto dos números reais
α	Letra grega Alfa
∞	Símbolo de infinito

Sumário

1	Introdução	23
1.1	Objetivos	24
1.1.1	Objetivo Geral	24
1.1.2	Objetivos Específicos	24
2	Fundamentação Teórica e Trabalhos Relacionados	27
2.1	Processamento de Imagens Digitais	27
2.1.1	Transformada Wavelet	27
2.1.1.1	Transformada de Wavelet Contínua	28
2.1.1.2	Transformada de Wavelet Discreta	28
2.1.2	Não Uniformidade da Foto-Reposta	29
2.1.3	Medidas de Qualidade de Imagens	30
2.1.4	Correlação	31
2.2	Aprendizado de Máquina	32
2.2.1	Redes Neurais Artificiais (RNA)	32
2.2.1.1	Modelo de Neurônio Biológico	33
2.2.1.2	Modelo de Neurônio Computacional	33
2.2.1.3	Perceptron	34
2.2.1.4	Retropropagação e gradiente descendente	34
2.2.1.5	Função de ativação	35
2.2.2	Aprendizagem Profunda	36
2.2.2.1	Redes Neurais Convolucionais	36
2.2.2.2	Convolução e camada convolucional	37
2.2.2.3	Pooling	38
2.2.3	Redes Adversárias Generativas (GANs)	39
2.2.4	Medidas de Avaliação	39
2.2.4.1	Matriz de Confusão	41
2.2.4.2	Top-N	42
2.3	A história do DeepFake	42
2.4	Trabalhos Relacionados	44
2.4.1	Criação de DeepFake	44
2.4.1.1	Primeiro Colocado	44
2.4.1.2	Segundo Colocado	45
2.4.2	Detecção de DeepFake	46
2.4.2.1	Primeiro colocado	46

2.4.2.2 Segundo colocado	47
3 Proposta do Trabalho	49
3.1 Método 1	49
3.1.1 Aquisição da Base de imagens	49
3.1.2 Detecção Facial	50
3.1.3 Recorte das imagens	51
3.1.4 Extração do PRNU	51
3.1.5 Treinamento da Rede Neural Convolucional	52
3.1.6 Experimentos 1 e 2	52
3.2 Método 2	52
3.2.1 Extração do PRNU	53
3.2.2 Utilização de Medidas de Qualidade da Imagem	54
3.2.3 Utilização da métrica Acurácia Top-N	54
3.2.4 Experimento 3	54
4 Resultados e discussão	55
4.1 Metodologia Experimental	55
4.1.1 Base de Imagens	55
4.1.2 PRNU utilizado na CNN	56
4.1.3 Rede Neural Convolucional	57
4.1.4 PRNU utilizado nos métodos de medida de qualidade da imagem	58
4.1.5 Métodos de medida de qualidade da imagem	58
4.2 Experimentos e Discussões	58
4.2.1 Experimento 1: classificação das imagens sem PRNU utilizando CNN .	58
4.2.2 Experimento 2: classificação das imagens com PRNU utilizando CNN .	58
4.2.3 Experimento 3: classificação das imagens PRNU utilizando acurácia top-N	59
4.2.4 Discussão geral	60
5 Conclusão	65
Referências	67

1 Introdução

A manipulação de imagens ou vídeos utilizando ferramentas de edição, como o Photoshop, sempre existiu, mas era muito mais simples poder realizar a detecção das imagens com os olhos humanos. Avanços recentes baseados em inteligência artificial e aprendizagem profunda permitiram a criação de conteúdos multimídias, de difícil reconhecimento ao olho humano, feitos por *DeepFake*. O método *DeepFake* pode criar imagens e vídeos falsos que são difíceis de verificar a autenticidade, utilizando da técnica de Redes Generativas Adversariais ([YU et al., 2021](#)).

O *DeepFake* possui diversas áreas para ser aplicado e inúmeras possibilidades no entretenimento, educação, jogos e acessibilidade. Segundo ([MIRSKY; LEE, 2021](#)), podemos utilizar *DeepFake* na dublagem de filmes estrangeiros, na educação através da reanimação de figuras históricas, e experimentando roupas virtualmente durante as compras. Além disso, existe uma comunidade disposta a criar "*memes*" utilizando *DeepFake*. Outro exemplo ocorreu em abril de 2019, com uma instituição benéfica na campanha chamada "*Malaria no More*". De acordo com ([SWATHI; SK, 2021](#)), foram criados vídeos e áudios, do ex futebolista David Beckham, utilizando *DeepFake*. Como parte dessa campanha social, Beckham falou em nove idiomas para transmitir a mensagem ao público.

No entanto, se for utilizada por pessoas erradas, pode se tornar uma prática antiética sendo utilizada para atividades maliciosas. Segundo ([SWATHI; SK, 2021](#)), o *DeepFake* pode criar vídeos pornográficos de celebridades utilizando a troca de faces ou criar discursos falsos para os líderes políticos no momento de eventos públicos. Qualquer pessoa com uma boa reputação ou tendo uma imagem pública pode ser alvo dessas atividades maliciosas. De acordo com ([YU et al., 2021](#)), o primeiro conteúdo *DeepFake* foi postado no *Reddit* em 2017 por um usuário chamado "*DeepFakes*". Nessa postagem havia um vídeo pornográfico de uma celebridade, o que já mostra como é inevitável que a tecnologia seja utilizada para o mal.

Logo após o primeiro caso, vários aplicativos baseados em *DeepFake* começaram a surgir como por exemplo *FakeApp* e *FaceSwap*. Outro exemplo segundo ([MIRSKY; LEE, 2021](#)), o *BuzzFeed* fez a divulgação de um vídeo *DeepFake* do ex-presidente Barack Obama dando uma palestra sobre o assunto. Esse vídeo foi feito com o software *FakeApp* e começou a levantar diversas preocupações sobre a falsificação de identidade e disseminação da desinformação nas mídias sociais. É também possível citar um caso malicioso ocorrido em 2019, onde surgiu um aplicativo chamado "*Deepnude*" que alterava digitalmente as imagens para remover roupas femininas, como consequência isso causou um enorme pânico em diversas pessoas ([SWATHI; SK, 2021](#)) ([YU et al., 2021](#)).

Como o *DeepFake* está se espalhando mais rápido do que o esperado e criando sérios

problemas, é necessário existir ferramentas e tecnologias automatizadas para detectar os conteúdos falsos. A detecção de *DeepFake* se tornou um dos principais problemas para as pessoas, as empresas e os governos em todo o mundo. E como consequência, diversas pesquisas relacionadas ao tema já estão em andamento, existindo até competições sobre o tema. Segundo ([SWATHI; SK, 2021](#)), recentemente a Amazon, Facebook e Microsoft uniram forças para criar o DeepFake Detection Challenge (DFDC) com o intuito de construir tecnologias inovadoras para detecção de imagens e vídeos falsos. Podemos também citar outras pesquisas sendo realizadas como por exemplo: o DeepFake Game Competition (DFGC) ou o programa *Media Forensics* da DARPA que também está trabalhando em tecnologias de detecção *DeepFake* ([YU et al., 2021](#))([PENG et al., 2021](#)).

Embora vários avanços já tenham sido alcançados, ainda existem muitos problemas para os métodos de detecção de *DeepFake*. Existe uma contínua evolução dos métodos *DeepFake*, onde os vídeos gerados se tornam cada vez mais realistas. Tendo isso em vista, esse trabalho realiza o estudo e implementação de duas técnicas de detecção de *DeepFake*. A primeira técnica realiza um recorte na face de cada imagem contida na base de dados e posteriormente é extraído o PRNU de cada imagem já recortada. A extração do PRNU é realizada para utilizar os padrões de ruídos residuais, deixados na imagem pela criação da GAN, no treinamento de uma CNN. A CNN aprenderá os padrões de ruído e conseguirá realizar a identificação das *DeepFakes*. A segunda técnica realiza novamente o recorte e a extração do PRNU, entretanto dessa vez será utilizado medidas de qualidade de imagem (Correlação, SSIM e MSE). Cada comparação realizada pelas medidas de qualidade produzem um coeficiente que diz se a imagem é similar ou não a outra, dessa maneira foi utilizado a Acurácia Top-*N* para realizar a detecção das *DeepFakes*.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo desse trabalho é desenvolver um método de detecção de *DeepFake*, utilizando recorte da face de uma imagem, extração do PRNU, treinamento de uma CNN com imagens PRNU, utilização de medidas de qualidade da imagem junto a acurácia top-N.

1.1.2 Objetivos Específicos

Visando atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

- Realizar um estudo sobre *DeepFakes*;
- Realizar um estudo e implementação de extração do PRNU;

- Realizar um estudo e implementação de arquiteturas de aprendizagem profunda;
- Realizar um estudo e implementação de métodos de qualidade da imagem;
- Desenvolver um protocolo experimental de validação;
- Analisar os resultados obtidos e discutí-los.

2 Fundamentação Teórica e Trabalhos Relacionados

2.1 Processamento de Imagens Digitais

O processamento de imagem é um método que realiza operações em uma imagem, buscando obter uma imagem aprimorada ou extrair alguma informação dela. A entrada é uma imagem e a saída pode ser uma imagem ou características associadas a essa imagem inicial ([QUEIROZ; GOMES, 2006](#)).

Segundo ([PEDRINI; SCHWARTZ, 2008](#)), uma imagem pode ser definida como uma função de intensidade luminosa, denotada por $f(x, y)$, cujo valor ou amplitude nas coordenadas espaciais (x, y) fornece a intensidade ou o brilho da imagem naquele ponto. Um modelo físico para a intensidade de uma cena pode ser expresso pelo produto de dois componentes, a quantidade de luz incidente e a quantidade de luz refletida. Esses componentes são chamados de iluminância e reflectância, e são representados por $i(x, y)$ e $r(x, y)$. De acordo com ([PEDRINI; SCHWARTZ, 2008](#)), a função $f(x, y)$ é dada por [2.1](#)

$$f(x, y) = i(x, y) \cdot r(x, y), \quad (2.1)$$

onde $0 < i(x, y) < \infty$ e $0 < r(x, y) < \infty$.

Uma imagem digital, pode ser obtida através de um processo chamado digitalização, o qual utiliza a amostragem e a quantização. De acordo com ([PEDRINI; SCHWARTZ, 2008](#)), a amostragem consiste em discretizar o domínio de definição da imagem na direção x e y , gerando uma matriz $M \times N$. Já a quantização consiste em escolher o número inteiro L permitido em cada ponto da imagem. Dessa maneira, cada elemento $f(x, y)$ existente na matriz é chamado de *pixel*, onde $0 \leq M - 1$ e $0 \leq N - 1$.

Segundo ([PEDRINI; SCHWARTZ, 2008](#)), uma imagem é monocromática quando pode ser descrita por uma função $f(x, y)$ da intensidade luminosa, equivalente aos níveis de cinza naquele pixel da imagem. Já uma imagem colorida é definida quando a cor em cada pixel $f(x, y)$ é definida através de três grandezas: luminância, matiz e saturação.

2.1.1 Transformada Wavelet

A Wavelet foi criada por Jean Morlet (Geofísico) e Alex Grossman (Físico) em 1981. Ela constitui uma nova base de estudo quando estamos relacionando sinais diversos, desde frequências elétricas até análise de sinais em previsões de eventos ([SILVA, 2014](#)).

Uma Wavelet é uma oscilação ondulatória localizada no tempo. As Wavelets possuem duas propriedades básicas: escala e localização. A escala define o quanto “esticada” uma wavelet é. Esta propriedade está relacionada com a frequência definida para as ondas. Já a localização define onde a wavelet está posicionada no tempo (ou espaço).

De acordo com ([SILVA, 2014](#)), para uma função ser denominada Wavelet são necessárias duas propriedades: a integral da função deve ser zero e a função deve ter energia unitária. Essas propriedades garantem respectivamente, que a função wavelet tenha uma forma de onda e possua um suporte compacto, garantindo uma localização temporal.

De forma geral, definimos as wavelets pela seguinte fórmula [2.2](#)

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}} \cdot \psi\left(\frac{t-b}{a}\right), \quad a, b \in \mathbb{R}, \quad a \neq 0, \quad (2.2)$$

onde a e b respectivamente, se referem a dilatação e a translação. A dilatação mede o grau de compressão e escala, já a translação determina a localização da wavelet no tempo ([SILVA, 2014](#)).

Existem duas versões da transformada de wavelet, a contínua e a discreta:

- Transformada de Wavelet Contínua (TWC): faz o mapeamento de uma função de uma variável contínua, em duas variáveis contínuas.
- Transformada de Wavelet Discreta (TWD) decompõe um sinal discreto em diferentes níveis de resolução.

2.1.1.1 Transformada de Wavelet Contínua

A TWC é baseada em uma única função $\Psi(t)$ igual a Transformada de Fourier, entretanto ela é deslocada gerando uma família de funções. Essa família pode ser definida por $\Psi_{a,b}(t)$.

De acordo com ([SILVA, 2014](#)), definimos a TWC como [2.3](#)

$$TWC(a, b) = \int_{-\infty}^{\infty} f(t) \Psi_{a,b}(t) dt, \quad (2.3)$$

onde $f(t)$ é transformado em uma nova função em um espaço bidimensional com escala a , e translação b pela transformada wavelet. O grupo de coeficientes $TWC(a,b)$ associados a um sinal $f(t)$, é a representação wavelet do sinal.

2.1.1.2 Transformada de Wavelet Discreta

Em processamento de sinais os dados podem ser representados por um conjunto finito de valores. Do ponto de vista matemático, uma função de dois parâmetros contínuos a e b ,

podem ser convertidos em uma representação discreta assumindo que a e b tenham somente valores inteiros ([SILVA, 2014](#)).

De acordo com ([SILVA, 2014](#)), definimos a TWD como [2.4](#)

$$TWD(m, n) = \frac{1}{\sqrt{a_0^m}} \int_{-\infty}^{\infty} f(x) \cdot \Psi(a_0^{-m} \cdot x - n \cdot b_0) dx, \quad (2.4)$$

onde m está relacionada ao escalonamento, e n está relacionado à translação. O parâmetro escala a é discretizado de forma exponencial $a = a_0^m$, enquanto o parâmetro b é discretizado proporcionalmente $a, b = b \cdot b_0 \cdot a_0^m$. As constantes a_0 e b_0 são comprimentos dos passos discretos de escala e translação respectivamente.

2.1.2 Não Uniformidade da Foto-Reposta

Abordagens baseadas em Padrão de Ruído dos Sensores para atribuição de fonte de imagem têm atraído atenção positiva da comunidade forense devido ao fato de serem uma maneira robusta para identificar a câmera específica, incluindo instâncias individuais do mesmo modelo, e não apenas a marca/modelo do dispositivo ([COSTA et al., 2012](#)). Podemos considerar dois tipos de padrões de ruído: Ruído de Padrão Fixo e Não Uniformidade da Foto-Reposta.

Quando uma quantidade fixa e uniforme de luz incide sobre as células do sensor em uma câmera digital, cada célula da câmera deveria produzir exatamente a mesma voltagem. No entanto, devido a uma variedade de fatores, incluindo pequenas variações no tamanho da célula e no material do substrato, isso acaba não sendo verdade. Quando uma luz uniforme brilha nas células de uma câmera digital, as células produzem voltagens ligeiramente diferentes. Essa diferença na resposta a uma fonte de luz uniforme é chamada de PRNU (Não Uniformidade da Foto-Reposta, do inglês, *Photo Response Non-Uniformity*). Como o PRNU é causado pelas propriedades físicas do próprio sensor, é quase impossível eliminá-lo. O PRNU é geralmente considerado uma característica normal do conjunto de sensores usado em uma câmera.

De acordo com ([COSTA et al., 2012](#)), o PRNU é definido por [2.5](#)

$$R_I = I - F(I), \quad (2.5)$$

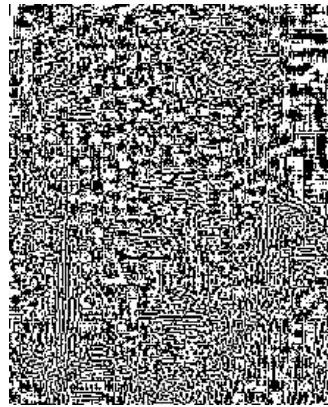
onde cada imagem I contido em um conjunto de imagens K , calcula o ruído residual R_i utilizando um filtro F baseado na Transformada de Wavelet Discreta (TWD).

Segundo ([COSTA et al., 2012](#)), as altas frequências (por exemplo, bordas de objetos) existentes em uma imagem podem contaminar seu componente PRNU e levar a resultados insatisfatórios de identificação por meio do padrão de ruído. Dessa maneira, é possível utilizar um método de aprimoramento do padrão de ruído (normalização) para reduzir a influência do conteúdo da cena. Segundo ([COSTA et al., 2012](#)), a normalização é dada por [2.6](#)

$$n_e(x, y) = \begin{cases} e^{-0.5n^2(x,y)/\alpha^2}, & \text{se } 0 \leq n^2(x, y); \\ -e^{-0.5n^2(x,y)/\alpha^2}, & \text{outro caso;} \end{cases} \quad (2.6)$$

onde cada pixel $n(x, y)$ gera um ruído aprimorado $n_e(x, y)$ e a constante α é definido pelo usuário. Pode se ver um exemplo da imagem PRNU normalizada na Figura 1.

Figura 1 – Exemplo de imagem PRNU.



Fonte: O Autor.

2.1.3 Medidas de Qualidade de Imagens

O objetivo das medidas de qualidade de imagens é comparar uma imagem X original a uma imagem Y estimada. Ou seja, deseja quantificar a comparação entre as imagens da mesma maneira em que a visão humana as compara. Realizando uma classificação de acordo com a similaridade existente entre as imagens.

O Mean Squared Error (MSE) mede a quantidade de erro em modelos estatísticos. Ele avalia a diferença quadrática média entre os valores observados e previstos. Quando um modelo não tem erro, o MSE é igual a zero. À medida que o erro do modelo aumenta, seu valor aumenta. Dois sinais são comparados pixel a pixel da esquerda para a direita e de cima para baixo através da linha e da coluna. Em seguida, é calculado a média quadrada da diferença entre o erro da imagem original e a imagem estimada.

Segundo ([GANDHI; KULKARNI, 2013](#)), o MSE é dado por [2.7](#)

$$MSE = \frac{1}{N} \sum_{i=1}^N |X_i - Y_i|^2, \quad (2.7)$$

onde N é a quantidade de previsões que serão realizadas, X_i é o valor a ser comparado e Y_i é o valor previsto.

O Structural Similarity Index (SSIM) irá procurar as semelhanças existentes nos pixels de diferentes imagens, realizando assim a comparação entre imagens. O principal objetivo do SSIM é extrair informações estruturais da imagem. Onde ele separa três parâmetros, como

luminância, contraste e estrutura, que são independentes uns dos outros e são altamente estruturados. O valor do SSIM varia entre -1 e 1 mas costuma se encontrar entre 0 e 1. Assume-se 1 para quando duas imagens idênticas, e 0 quando duas imagens completamente diferentes são dadas como entrada.

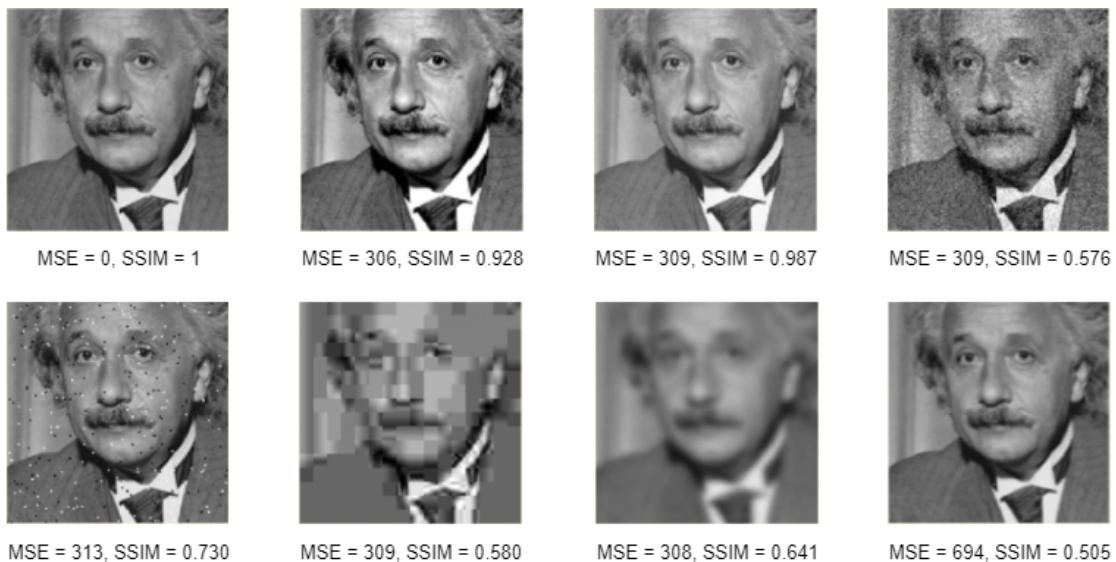
Segundo ([GANDHI; KULKARNI, 2013](#)), o SSIM é dado por [2.8](#)

$$SSIM(X, Y) = \frac{(2\mu_X\mu_Y + C1)(2\sigma_X\sigma_Y + C2)}{(\mu_X^2 + \mu_Y^2 + C1)(\sigma_X^2 + \sigma_Y^2 + C2)}, \quad (2.8)$$

onde μ_X e μ_Y representam a média de X e Y , respectivamente; σ_X e σ_Y representam as variâncias de X e Y , respectivamente; $\sigma_X\sigma_Y$ representa a covariância entre os dois sinais e $C1$ e $C2$ são constantes que estabilizam a divisão, caso exista um denominador próximo de zero.

Podemos ver um exemplo de uso do MSE e SSIM na Figura [2](#).

Figura 2 – Comparação dos métodos SSIM e MSE para imagens de Einstein alteradas com diferentes tipos de distorções.



Fonte: ([WANG; BOVIK, 2009](#)).

2.1.4 Correlação

A correlação é uma medida estatística que expressa até que ponto duas variáveis estão linearmente relacionadas, ou seja, a força de associação entre as duas variáveis. Essa é uma ferramenta comum para descrever relacionamentos simples sem fazer uma declaração sobre causa e efeito. Uma correlação positiva indica até que ponto essas variáveis aumentam ou diminuem juntas. Já uma correlação negativa indica até que ponto uma variável aumenta à medida que a outra diminui.

Em estatística, o valor do coeficiente de correlação varia entre +1 e -1. Segundo ([ZAIK, 2015](#)), quando o valor do coeficiente de correlação fica em torno de ± 1 , diz-se que

é um grau perfeito de associação entre as duas variáveis. À medida que o valor do coeficiente de correlação se aproxima de 0, a relação entre as duas variáveis será mais fraca. Normalmente, em estatística, é medido três tipos de correlações: correlação de Pearson, correlação de Kendall e correlação de Spearman. Neste trabalho foi utilizado apenas a correlação de Pearson.

Correlação de Pearson: é amplamente utilizada em estatística para medir o grau de relação entre variáveis lineares relacionadas. Por exemplo, no processamento de imagens, se quisermos medir como o ruido residual de duas imagens estão relacionadas entre si, a correlação de Pearson é usada para medir o grau de relacionamento entre os dois ruidos residuais.

Segundo ([COSTA et al., 2012](#)), a função 2.9 é dada por

$$\text{corr}(I_a, I_b) = \frac{(I_a - \bar{I}_a) \cdot (I_b - \bar{I}_b)}{\|I_a - \bar{I}_a\| \cdot \|I_b - \bar{I}_b\|} \quad (2.9)$$

onde corr é o valor da correlação entre as duas imagens I_a e I_b ; \bar{I}_a e \bar{I}_b são as médias das imagens I_a e I_b respectivamente; $\|I_a - \bar{I}_a\|$ e $\|I_b - \bar{I}_b\|$ são as normas da diferença entre as imagens e suas médias.

2.2 Aprendizado de Máquina

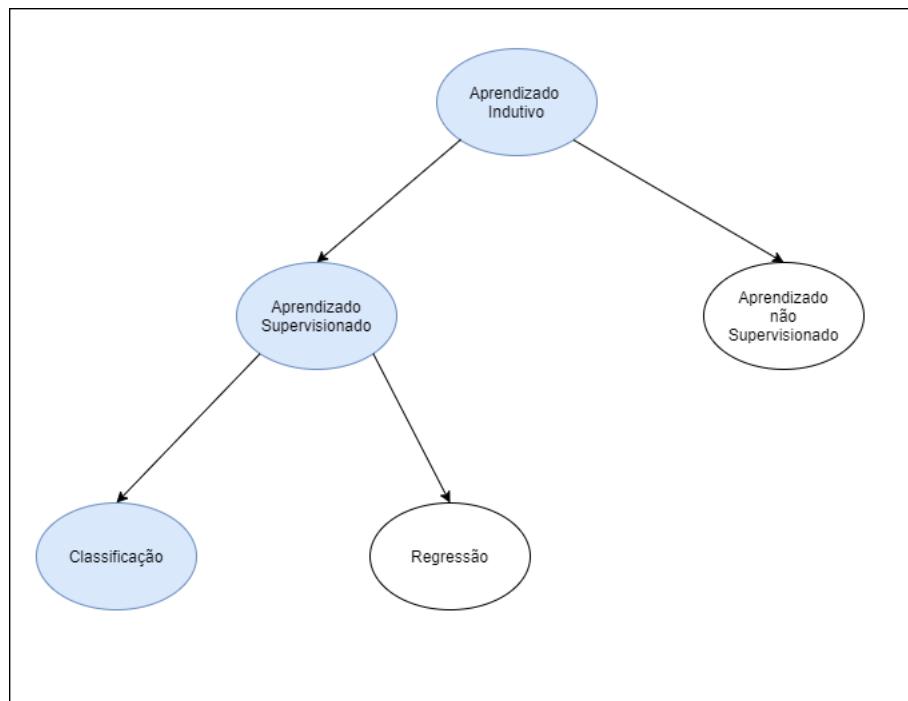
Aprendizado de Máquina (AM) é uma área de Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Dentro de AM existem técnicas que podem ser divididas em aprendizado supervisionado e não-supervisionado. ([MONARD; BARANAUSKAS, 2003](#)).

No caso de aprendizado supervisionado, é dado ao algoritmo um conjunto de exemplos com atributos de entrada e atributos de saídas (rótulos). O objetivo do algoritmo é construir um classificador que pode determinar corretamente o rótulo de um novo exemplo. Para valores discretos esse problema é conhecido como classificação, já para valores contínuos, ele é conhecido como regressão. Em contraste, no aprendizado não-supervisionado é dado um conjunto de exemplos apenas com atributos de entrada, onde o algoritmo analisa e tenta determinar se eles podem ser agrupados de alguma maneira. Podemos ver um exemplo da hierarquia do aprendizado na Figura 3.

2.2.1 Redes Neurais Artificiais (RNA)

As Redes Neurais Artificiais (RNAs) são algoritmos de aprendizagem de máquina que simulam o funcionamento das redes de neurônios biológicos. De acordo com ([HAYKIN, 2001](#)), o cérebro tem a capacidade de organizar seus constituintes estruturais (neurônios) de forma a realizar certos processamentos (reconhecimento de padrões, percepção e controle motor, entre outros) de maneira muito mais rápida e eficiente que um supercomputador.

Figura 3 – Hierarquia do Aprendizado.



Fonte: O Autor

2.2.1.1 Modelo de Neurônio Biológico

De acordo com ([BIANCHINI, 2001](#)), o modelo de Neurônio Biológico, conforme verificado na Figura 4, é composto por 4 partes:

- Dendrites: Faz a coleta dos impulsos oriundos de outros neurônios;
- Corpo Celular: Responsável por processar os sinais recebidos pelas dendrites;
- Axônio: Encarregado pela multiplicação dos sinais;
- Sinapse: Conexão do axônio de um neurônio com a dendrite de outro;

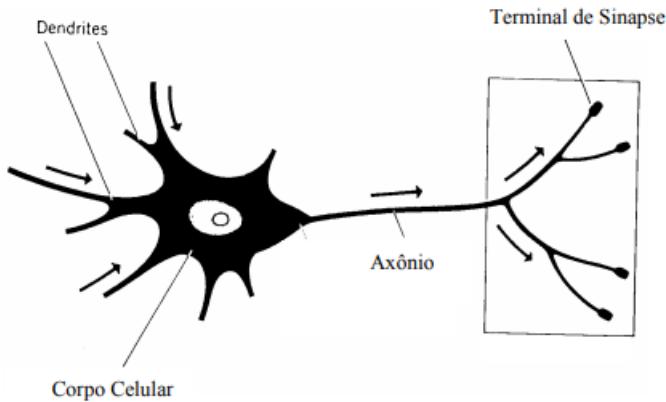
2.2.1.2 Modelo de Neurônio Computacional

O modelo computacional possui características semelhantes ao do modelo biológico. Toda rede neural possui diversos neurônios, e de acordo com ([HAYKIN, 2001](#)) "Um neurônio é uma unidade de processamento de informação que é fundamental para a operação de uma rede neural".

Identificamos três elementos básicos de um modelo neural.

- Conjunto de sinapses ou elos de conexão: Como foi visto anteriormente, a sinapse é responsável por ligar um neurônio a outro. Nesse caso, ela também serve para passar

Figura 4 – Neurônio Biológico



Fonte: ([BIANCHINI, 2001](#))

informação. Ela transmite para o próximo neurônio o peso sináptico e o sinal de entrada, multiplicando ambos. Dessa maneira, ela consegue atribuir um valor ao próximo neurônio e tratar de maneira distinta cada informação.

- Somador: Cada neurônio recebe K sinapses. Ele é responsável por realizar a somatória de todas as sinapses recebidas. Ele também é conhecido como Bias.
- Função de ativação ou função restritiva: ela é utilizada para restringir a amplitude do valor de saída de um neurônio a um valor finito.

2.2.1.3 Perceptron

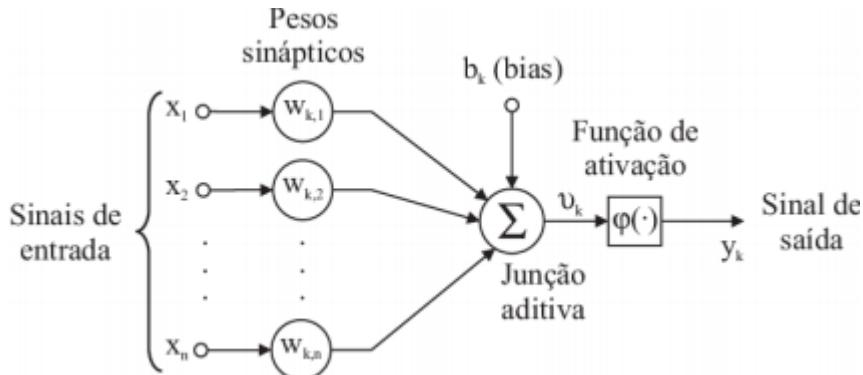
O perceptron é a rede neural mais simples que existe, ela possui apenas um neurônio. De acordo com ([BIANCHINI, 2001](#)), o perceptron é um modelo de rede neural utilizado para a classificação de padrões linearmente separáveis. Segundo ([RODRIGUES, 2018](#)), um Perceptron realiza o somatório do produto escalar $\chi_i \cdot \omega_i$ adicionando a ele um bias b, onde χ_i são as suas entradas e ω_i são seus respectivos pesos associados. Após esse valor ser processado, ele é enviado para uma função de ativação e é retornado o valor 0 ou 1 como resposta. O perceptron sempre retornará uma resposta binária. O *perceptron* é representado pela Figura 5.

O perceptron é uma rede do tipo *feedforward*, ou seja, ela sempre propaga em apenas uma direção. Geralmente visualizamos da esquerda para a direita. O *input* entra pelo perceptron e o sinal se propaga para a saída, sem que ele retorne para o começo.

2.2.1.4 Retropropagação e gradiente descendente

As RNAs aprendem através do ajuste de peso em cada camada. Entretanto, para definir se esses pesos estão corretos, é necessário definir uma função de perda. De acordo com ([RODRIGUES, 2018](#)), ela se faz necessária pois mede a imprecisão da rede dado um resultado

Figura 5 – Perceptron.



Fonte: ([BIANCHINI, 2001](#)).

esperado. Uma função de perda muito utilizada é a *cross-entropy*. Segundo ([KOECH, 2020](#)), a função *cross-entropy* é dada por [2.10](#)

$$P = - \sum_{i=1}^n t_i \log(p_i), \quad (2.10)$$

onde P é o resultado da função de perda, n é a quantidade de classes, t_i é a classe verdadeira e p_i é a probabilidade de acerto para a i -ésima classe.

Para atingir os resultados esperados é necessário ajustar os pesos utilizando um algoritmo, um dos quais mais utilizados para isto é a retropropagação (backpropagation) com o gradiente descendente. De acordo com ([VIEIRA et al., 2017](#)), essa combinação permite obter o gradiente para a sequência de parâmetros presentes na rede usando a regra da cadeia.

Segundo ([BIANCHINI, 2001](#)), a retropropagação se trata de uma rede supervisionada que requer uma resposta desejada para ser treinada onde a perda para cada neurônio de saída é calculado comparando o valor obtido para a saída com o valor alvo. Após essa perda ser calculada, ela é passada para as próximas camadas, onde é possível ponderar os erros das camadas superiores. Ela altera os pesos w para possibilitar que a rede aprenda os padrões dos dados.

Como visto, os valores dos pesos são alterados pela retropropagação, que se chama taxa de aprendizagem. Com isso, segundo ([RODRIGUES, 2018](#)), o objetivo do gradiente descendente é minimizar a função de perda para a menor perda local, necessitando de um cuidado na escolha da taxa de aprendizado.

Esses dois algoritmos atuam na rede *feedforward*, onde ele calcula a perda na ida e os pesos são ajustados na volta.

2.2.1.5 Função de ativação

A função de ativação é uma parte importante da RNA, sendo ela que determina o resultado de cada neurônio. Se funções lineares forem utilizadas ela produz soluções convexas.

Entretanto para solução não convexas, é necessário o uso de funções não-lineares, segundo (RODRIGUES, 2018). Existem diversas funções de ativação, as principais utilizadas são a sigmoid e a RELU.

- Sigmoid: também conhecida como função logística, é dada pela seguinte equação

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.11)$$

- RELU: segundo (AGARAP, 2018), RELU é uma função de ativação, que tem forte embasamento biológico e matemático. Em 2011, demonstrou-se que melhorava ainda mais o treinamento de redes neurais profundas. Ele funciona limitando os valores em 0, ou seja, $f(x) = \max(0, x)$. Simplificando, ele produz 0 quando $x < 0$ e, inversamente, ele produz uma função linear quando $x \geq 0$.

2.2.2 Aprendizagem Profunda

A aprendizagem profunda (*Deep Learning*) é um algoritmo de aprendizado de máquina, que possibilita obter um grau de exatidão maior com problemas mais complexos. Essa técnica vem sendo utilizada em diversas áreas de estudo como por exemplo: robótica, reconhecimento imagem, áudio, reconhecimento facial, entre outras aplicações (VIEIRA et al., 2017).

A Aprendizagem Profunda surgiu para tratar a dificuldade que arquiteturas comumente utilizadas, como RNAs ou Support Vector Machines (SVMs) possuem em dados com alta dimensionalidade, segundo (ARNOLD et al., 2011). Este é proporcional ao volume de espaço em que os dados são inseridos, sendo mais dispersos quanto maior a dimensionalidade.

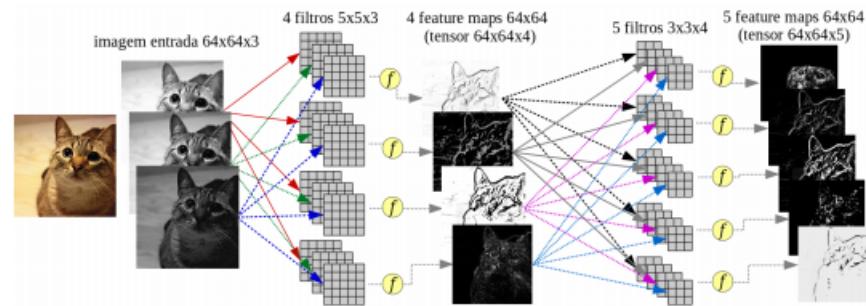
2.2.2.1 Redes Neurais Convolucionais

Para (VIEIRA et al., 2017), as Redes Neurais Convolucionais (CNNs) são provavelmente os modelos de *Deep Learning* mais conhecidos e utilizados atualmente. Esse tipo de rede neural utiliza camadas convolucionais, para conseguir processar a entrada, considerando campos receptivos locais. A convolução é parte do processo da CNN. De acordo com (GOODFELLOW; BENGIO; COURVILLE, 2016), ela é feita em 3 partes: a convolução de cada camada de entrada; a aplicação de uma função de ativação não linear e outra de subamostragem (pooling). Com isso, a etapa de convolução caracteriza-se pela passagem do núcleo (kernel) pela imagem (input) e o resultado desse processamento é denominado mapa de características (output), o qual permitirá o reconhecimento de padrões, inclusive de outros padrões da rede.

Algumas particularidades da CNN, faz com que ela seja diferente da rede neural padrão. Dessa maneira, ela reduz uma grande quantidade de parâmetros gerados pela sua computação. Segundo (GOODFELLOW; BENGIO; COURVILLE, 2016), elas são: conectividade esparsa das unidades de processamento; campo receptivo local; arranjo espacial; compartilhamento de pesos e *dropout*.

Uma CNN possui algumas camadas, incluindo um ou mais planos. A imagem é inserida na primeira camada, onde cada filtro (neurônio) dessa camada irá processar essa imagem e transforma-la através de combinações lineares com pixels vizinhos. Cada filtro já possui pesos estabelecidos para que consigam obter uma mesma característica. Assim vários filtros são utilizados em múltiplas camadas, para que todas as características possam ser encontradas. Podemos ver um exemplo na Figura 6.

Figura 6 – Exemplo de camadas convolucionais



Fonte: ([VIEIRA et al., 2017](#))

2.2.2.2 Convolução e camada convolucional

Segundo ([RODRIGUES, 2018](#)), a convolução é um operador linear que recebe duas funções como entrada e retorna uma terceira função. O retorno é originário do somatório da multiplicação da função kernel na região superposta da função alvo pelo deslocamento do kernel sobre ela. A formula da convolução é dada por 2.12

$$(f \cdot g)(k) = h(k) = \sum_{i=0}^k f(i) \cdot g(k-i), \quad (2.12)$$

onde f e g são sequências numéricas do tamanho. Os kernels podem também ser considerados filtros de convolução. Esses filtros percorrem a imagem e calculam o produto escalar, onde cada filtro extrai diferentes recursos da imagem.

A camada convolucional de uma RNA é responsável por extrair as características de entrada. De acordo com ([VIEIRA et al., 2017](#)) na camada convolucional cada neurônio é um filtro aplicado a uma imagem de entrada e cada filtro é uma matriz de pesos.

O processo de extração de características é dado por meio de filtros convolucionais, onde os filtros percorrem os dados de entrada realizando a convolução sobre os dados. A cada processamento durante o treinamento, os filtros são ajustados de acordo com características comuns em lotes de entrada.

Podemos considerar um filtro de tamanho 3×3 e uma imagem de tamanho 5×5 como exemplo. Realizando uma multiplicação por elementos entre os valores de pixel da imagem que

correspondem ao tamanho do kernel e o próprio kernel, e por fim somamos todos os valores. Isso nos fornece um único valor para a célula de característica. Isso é representado na Figura 7, onde foi realizado o seguinte cálculo: $2 \cdot 1 + 4 \cdot 2 + 9 \cdot 3 + 2 \cdot (-4) + 1 \cdot 7 + 4 \cdot 4 + 1 \cdot 2 + 1 \cdot (-5) + 2 \cdot 1 = 51$ para chegar ao resultado da célula de característica.

Figura 7 – Exemplo de convolução

2	4	9	1	4
2	1	4	4	6
1	1	2	9	2
7	3	1	5	3
2	3	8	5	4

X

1	2	3
-4	7	4
2	-5	1

=

51		

Imagen
Kernel ou Filtro
Característica

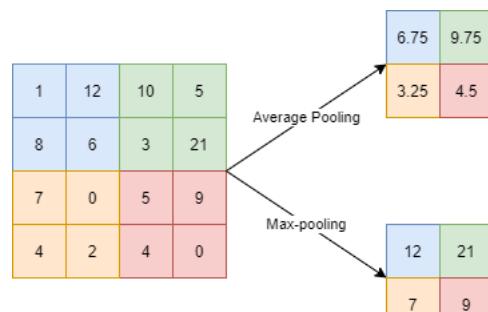
Fonte: O Autor.

2.2.2.3 Pooling

O *pooling* consiste em uma camada que reduz o tamanho de entrada dos dados. Geralmente após uma camada convolucional é utilizado uma camada de *pooling* de tal forma que as próximas camadas recebam uma outra forma de representação dos dados. Segundo (SANTANA, 2019), esta camada seleciona uma região do mapa de características através de um filtro e aplica uma subamostragem, sendo os tipos mais comuns *average pooling* e *max-pooling*.

De acordo com (GHOLAMALINEZHAD; KHOSRAVI, 2020), a *average pooling* é a maneira para agrupar e extrair as características por meio da média dos valores de uma determinada região do mapa de características. É a primeira rede neural profunda baseada em convolução. Já no *max-pooling* é utilizado o maior valor dentro da região. Na Figura 8 é possível ver um exemplo do *average pooling* e do *max-pooling*.

Figura 8 – Exemplo de aplicação do *max-pooling* e *average pooling* de tamanho 2×2 com passo 2 sobre uma mapa de características



Fonte: O autor

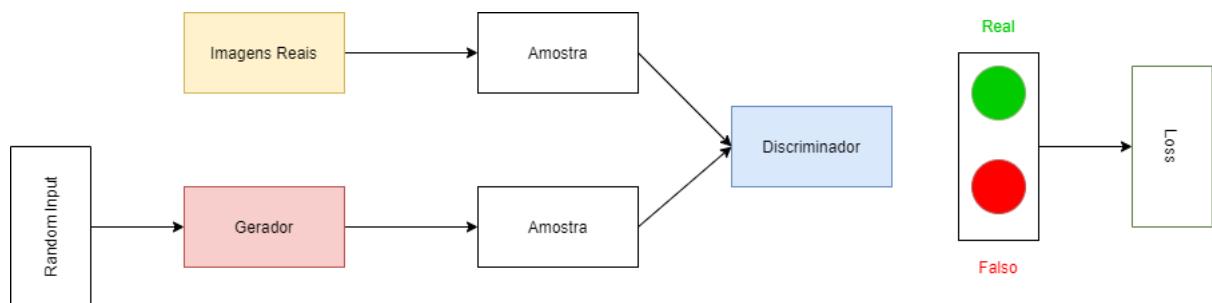
2.2.3 Redes Adversárias Generativas (GANs)

Segundo (SANTOS et al., 2020), as Redes Adversárias Generativas são fundamentadas em uma arquitetura com o propósito de construir modelos gerativos, que por sua vez tentam aprender a distribuição de dados.

De acordo com (GOODFELLOW et al., 2014), as GANs são feitas por uma estrutura onde duas redes neurais aprendem juntas por meio de uma competição. Existe a rede Discriminadora, que realiza o papel de identificar se um objeto faz parte de uma classe específica e a rede Geradora, responsável por fornecer uma entrada feita com valores aleatórios que gera um objeto para ser avaliado pela rede Discriminadora.

Segundo (GOODFELLOW et al., 2014), essa dinâmica entre as redes funciona como um jogo de minimax de dois jogadores. Nessa competição entre ambas as redes, a rede Geradora cria objetos cada vez mais difíceis de serem distinguidos pela rede Discriminadora. Com esse aumento na dificuldade, a rede Discriminadora também se aperfeiçoa para responder se um objeto pertence ou não a uma classe. Temos um exemplo na Figura 9.

Figura 9 – Exemplo de GAN



Fonte: Autor

Essa aplicação pode ser usada em diversas áreas, como por exemplo: geração de imagens realistas à partir de textos, melhora da qualidade de uma imagem ou criação de *DeepFakes*. Pode-se citar a StyleGAN como exemplo de GAN muito poderosa para a criação de imagens realistas. A Figura 10 mostra dois exemplos, uma imagem real e outra falsa criada pelo método StyleGAN, as quais mostram o poder que a StyleGAN possui na tarefa de criação de imagens falsas.

2.2.4 Medidas de Avaliação

Segundo (MONARD; BARANAUSKAS, 2003), dado um conjunto de exemplos de tamanho finito e um indutor, é importante estimar o desempenho futuro do classificador induzido utilizando o conjunto de exemplos. O mundo real possui uma distribuição D desconhecida de exemplos em um dado domínio. Ao retirar alguns exemplos do mundo real, é obtido uma distribuição D' , a qual é supostamente similar à distribuição D . Para estimar uma medida, geralmente a precisão ou o erro, de indutores treinados com base na distribuição D' , extraem-se amostras a

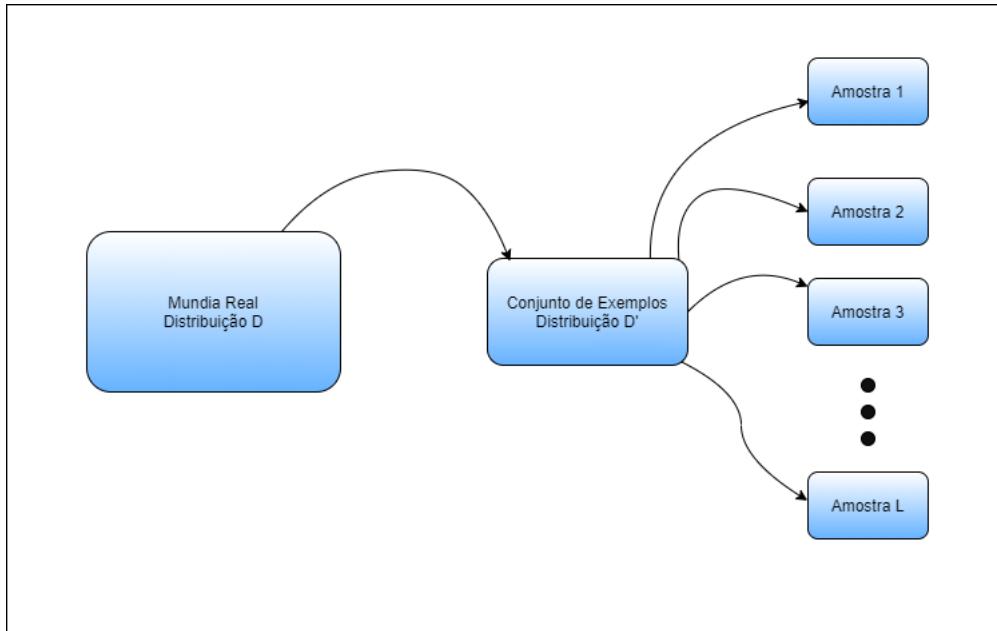
Figura 10 – Exemplo de imagem real e imagem criada pela StyleGAN.



Fonte: O Autor

partir de D' , treina-se um indutor com essas amostras e testa-se seu desempenho em exemplos de D' . Dessa maneira, é possível simular o processo de amostragem que ocorre no mundo real.

Figura 11 – Técnicas de estimativas baseadas na idéia de amostragem



Fonte: O Autor.

É importante, ao estimar uma medida verdadeira, que a amostra seja aleatória. Para problemas reais, normalmente é feita uma amostra de tamanho n e o objetivo consiste em estimar uma medida para aquela população em particular.

Para se obter os resultados primeiramente é necessário ser feito uma matriz de confusão e através dela é retirado alguns resultados.

2.2.4.1 Matriz de Confusão

A matriz de confusão oferece uma medida efetiva do modelo de classificação, ao mostrar o número de classificações corretas versus as classificações preditas para cada classe, em um conjunto de exemplos T . Os resultados são totalizados em duas dimensões: classes verdadeiras e classes preditas, para k classes diferentes $\{C_1, C_2, \dots, C_k\}$. Cada elemento $M(C_i, C_j)$ da matriz, é dado por 2.13

$$M(C_i, C_j) = \sum_{\{\forall(x,y) \in T : y = C_i\}} \|h(x) = C_j\|, \quad (2.13)$$

representa o número de T que realmente pertencem à classe C_i , mas foram classificados como sendo da classe C_j .

Tabela 1 – Matriz de confusão de um classificador

Classe	predita C_1	predita C_2	...	predita C_k
verdadeira C_1	$M(C_1, C_1)$	$M(C_1, C_2)$...	$M(C_1, C_k)$
verdadeira C_2	$M(C_2, C_1)$	$M(C_2, C_2)$...	$M(C_2, C_k)$
\vdots	\vdots	\vdots	\ddots	\vdots
verdadeira C_k	$M(C_k, C_1)$	$M(C_k, C_2)$...	$M(C_k, C_k)$

O número de acertos, para cada classe, se localiza na diagonal principal $M(C_i, C_i)$ da matriz. Os demais elementos $M(C_i, C_j)$, para $i \neq j$, representam erros na classificação. A matriz de confusão ideal possui todos esses elementos iguais a zero já que ele não comete erros.

Considere um problema com duas classes. Com apenas duas classes, rotularemos como “+” (positivo) e “-” (negativo), as escolhas estão estruturadas para predizer a ocorrência ou não de um evento simples. Nesse caso, os dois erros possíveis são denominados *falso positivo* (F_P) e *falso negativo* (F_N). Na Tabela 2, a matriz de confusão é ilustrada para o problema com duas classes, onde V_P é o número de exemplos positivos classificados corretamente e V_N é o número de exemplos negativos classificados corretamente do total de $n = (V_P + V_N + F_P + F_N)$ exemplos.

Tabela 2 – Matriz de confusão para classificação com duas classes

Classe	predita C_+	predita C_-	Taxa de erro da classe	Taxa de erro total
verdadeiro C_+	Verdadeiros positivos V_P	Falsos negativos F_N	$\frac{F_N}{V_P + F_N}$	$\frac{F_P + F_N}{n}$
	Falsos positivos F_P	Verdadeiros negativos V_N		
verdadeiro C_-			$\frac{F_P}{F_P + V_N}$	$\frac{F_P + F_N}{n}$

Segundo (JOST, 2015), através do resultado da matriz de confusão são utilizadas 4 fórmulas para realizar a avaliação do classificador. Essas fórmulas são Acurácia, Precisão, Revocação e F1-Score respectivamente.

- Acurácia: efetua uma relação entre amostras classificadas como corretas e o total de amostras;

$$acc(h) = \frac{V_P + V_N}{n} \quad (2.14)$$

- Precisão: determina o quanto o classificador é capaz de rejeitar as outras classes;

$$prec(h) = \frac{V_P}{V_P + F_P} \quad (2.15)$$

- Revocação: capacidade de classificação de amostras pertencentes à classe analisada;

$$rec(h) = \frac{V_P}{V_P + F_N} \quad (2.16)$$

- F1-Score: é a medida harmônica das medidas de precisão e recall.

$$f1(h) = \frac{2 \times (prec(h) \times rec(h))}{(prec(h) + rec(h))} \quad (2.17)$$

2.2.4.2 Top-N

Muitas vezes, ao construir modelos de aprendizado de máquina, focamos na métrica de acurácia, tentando obter a classe certa de uma imagem. Mas quando se utiliza a acurácia apenas para prever a probabilidade mais alta, uma limitação pode ser causada nas áreas às quais a técnica pode ser aplicada. Dessa maneira, é possível também utilizar a acurácia top- N .

A acurácia top- N leva as N previsões do modelo com maior probabilidade. Se um deles for um rótulo verdadeiro, ele classifica a previsão como correta. A acurácia top-1 é um caso especial, no qual apenas a previsão de probabilidade mais alta é levada em consideração. Podemos utilizar como exemplo para acurácia top-3 a tabela 3, que representa um modelo, o qual tenta classificar imagens de animais. Este modelo possui uma previsão de 80% de acerto utilizando a acurácia top-3, ou seja, o modelo acertou 4/5 imagens.

Tabela 3 – Exemplo de acurácia top-3

Classe Real	Classe Predit	Correta
Cachorro	Cachorro , Gato, Girafa	✓
Gato	Cachorro, Gato , Passaro	✓
Passaro	Gato, Golfinho, Girafa	✗
Golfinho	Passaro, Cachorro, Golfinho	✓
Girafa	Girafa , Cachorro, Passaro	✓

Fonte: O Autor

2.3 A história do DeepFake

A manipulação facial não é uma tecnologia que apareceu recentemente. Segundo (YU et al., 2021), a primeira tentativa de manipulação facial na literatura pode ser encontrada no icônico retrato de 1865 do presidente dos EUA, Abraham Lincoln. Com a evolução da computação

gráfica, a manipulação facial em imagens digitais acabou se tornando alcançável. Porém a evolução não parou, outro progresso recente foi alcançado no campo da aprendizagem profunda, onde houve um avanço fundamental no desenvolvimento da manipulação facial.

Tecnologias baseadas em aprendizagem profunda permitiram a criação de textos, áudios, vídeos e fotografias falsas, todos os quais podem à primeira vista parecer genuínos. A IA demonstrou uma recente capacidade para criar vídeos realistas, pois é capaz de pegar um vídeo já existente e sobrepor um rosto a uma fotografia, ou alterar a voz fazendo com que alguém diga coisas que não são realidade e nunca foram realizados ([KARNOUSKOS, 2020](#)). Essa tecnologia têm um grande potencial para remodelar a mídia digital, enquanto as implicações causadas a sociedade também podem ser graves. Essas ações causam uma desconfiança ao público no conteúdo visto e ouvido. De acordo com ([NGUYEN et al., 2019](#)), os *DeepFakes* se tornaram populares devido à sua qualidade em criar vídeos adulterados, além de tornar fácil o seu uso em diversos aplicativos. Uma ampla gama de usuários pode utilizar desse recurso sem precisar ter nenhum conhecimento na área.

Segundo ([NGUYEN et al., 2019](#)), a primeira tentativa de criação *DeepFake* foi o *FakeApp*, desenvolvido por um usuário do *Reddit* usando um auto codificador. Nesse método, o codificador automático extrai características latentes de várias imagens do rosto de pessoas e o decodificador é usado para reconstruir essas imagens de rosto. Para trocar faces entre imagens de origem e destino, há uma necessidade de dois pares codificador-decodificador onde cada par é usado para treinar em um conjunto de imagens, e os parâmetros do codificador são compartilhados entre as duas redes. Ou seja, é utilizado a estrutura de GANs para realizar as *DeepFakes*.

Após o primeiro caso ocorrer, vários *DeepFakes* começaram a ficar populares e circular na Internet. Esses *DeepFakes* podem ser encontrados em diversos sites populares como por exemplo no YouTube. Um dos mais famoso *DeepFake* encontrados na internet é o vídeo de 2018 do Barack Obama, onde ironicamente, ele faz um alerta sobre os perigos do *DeepFake*. Alguns outros *DeepFakes* são vídeos pornôs de celebridades, bem como uma deturpação de políticos, por exemplo, substituir o rosto de Angela Merkel por Donald Trump, ou Donald Trump por Mr. Bean ([KARNOUSKOS, 2020](#)).

De acordo com ([YU et al., 2021](#)), os métodos de *DeepFake* existentes podem ser divididos em duas categorias: troca da face e reconstituição facial. Com base no método de trocar de face, a face recém renderizada pode ser composta no vídeo de origem, substituindo a face original. Em dezembro de 2017, o primeiro vídeo de troca da face gerado por *DeepFake* foi postado por um usuário do *Reddit*. Segundo ([YU et al., 2021](#)), é reconhecido que a inspiração do método *DeepFake* vem do trabalho "Mask R-CNN" de ([HE et al., 2017](#)), onde as CNNs foram utilizadas para gerar imagens de troca de face. Depois dessa primeira aparição, uma onda de criação de vídeos utilizando a troca da face começou a se espalhar pelo mundo. Outros exemplos de trabalhos que utilizaram essa técnica foram: Faceswap-GAN, VGGFace e DeepFaceLab.

Diferente das tecnologias de troca de rostos, os métodos de reconstituição facial tentam controlar as expressões das pessoas nos vídeos, o que significa que os invasores podem gerar vídeos manipulando alguém para fazer algo que não existe (YU et al., 2021). O primeiro surgimento dessa técnica utilizada em *DeepFake* foi o trabalho "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition" feito por (THIES et al., 2016) onde foi proposto o Face2Face. Segundo (YU et al., 2021), esse trabalho propôs uma nova abordagem global de agrupamento baseado em modelo não rígido, que foi aplicada para reconstruir as características faciais de atores alvo e fonte. Porém também é utilizada uma técnica de deformação espacial ao mesmo tempo, a qual realiza uma transferência de expressão entre os atores alvo e fonte. Infelizmente, essa técnica não pode garantir que movimentos da cabeça sejam consistentes, pois apenas a migração das expressões é levada em consideração. A boca é outro ponto não satisfatório, pois os detalhes grosseiros da boca podem ser facilmente percebidos pelos olhos humanos. Com esses problemas outros trabalhos foram criados para tentar melhorar a técnica de reconstituição facial, de acordo com (YU et al., 2021), podemos citar os trabalhos "Synthesizing obama: learning lip sync from audio" feito por (SUWAJANAKORN; SEITZ; KEMELMACHER-SHLIZERMAN, 2017), "Text-based editing of talking-head video" feito por (FRIED et al., 2019) e "Deep video portraits" feito por (KIM et al., 2018).

2.4 Trabalhos Relacionados

Esta seção será destinada a discussão sobre os trabalhos relacionados apresentados no artigo "DFGC 2021: A DeepFake Game Competition" feito por (PENG et al., 2021). Esse artigo apresenta um resumo da competição DFGC 2021, apresenta os melhores resultados da competição e também disponibiliza o conjunto de dados feito pelos participantes (sendo que essa base de dados foi utilizada em nosso trabalho).

Serão citados os 2 melhores métodos de *DeepFake* e os 2 melhores métodos de detecção de *DeepFake* da competição, que foram apresentados no artigo (PENG et al., 2021).

2.4.1 Criação de DeepFake

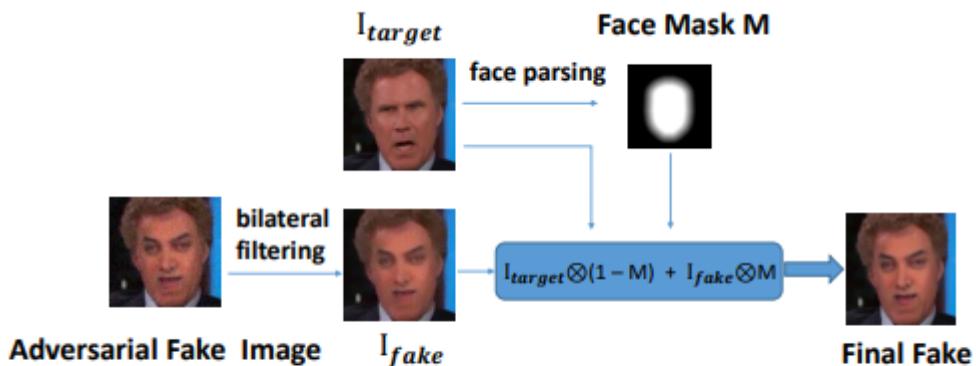
2.4.1.1 Primeiro Colocado

De acordo com (PENG et al., 2021), o primeiro colocado escolheu o método *faceshifter* para gerar as imagens falsas. Com base no modelo de *faceshifter* (BUSLAEV et al., 2020), foram adotadas três estratégias para melhorar a pontuação. A primeira estratégia foi ajustar o modelo *faceshifter*, a segunda foi adicionar ruído adversário à imagem gerada e a terceira foi realizar um método de pós-processamento na imagem gerada, para reduzir a área de fraude e o nível de ruído na imagem final.

Neste trabalho, a operação de troca de faces envolveu duas imagens, a de origem e a de

destino. Para obter um melhor efeito de troca de rosto, foi treinado um modelo exclusivo para cada par ($Imagen_{fonte}$, $Imagen_{alvo}$). Para cada par, foram escolhidos os vídeos correspondentes a $Imagen_{fonte}$, e os vídeos correspondentes a $Imagen_{alvo}$ para criar um subconjunto de dados. Em seguida, o subconjunto de dados é utilizado para ajustar o modelo pré-treinado do *faceshifter*. Dessa maneira as imagens geradas ficaram mais limpas, aumentando a pontuação da equipe na competição, entretanto mais alterações foram realizadas. O próximo passo foi adicionar ruídos adversariais as imagens geradas. Para isso o grupo treinou um detector utilizando a o conjunto de dados Celeb-DF e as imagens que eles já aviam gerado, o qual usava a arquitetura EfficientNet-B7. Para finalizar eles realizaram um pós-processamento na imagem, onde foi utilizado um filtro bilateral nas imagens. Após foi fundido a região da face da imagem falsa na imagem alvo para gerar a imagem final. A figura 12 mostra os passos que foram seguidos pelo trabalho.

Figura 12 – Passos seguidos pelo primeiro colocado na geração de *DeepFake*.



Fonte: ([PENG et al., 2021](#)).

2.4.1.2 Segundo Colocado

De acordo com ([PENG et al., 2021](#)), o segundo colocado utilizou um modelo *FaceShifter* pré-treinado, como o modelo professor que orienta a aprendizagem de um aluno *FaceShifter*, e utiliza um método de treinamento adversarial para melhorar a capacidade de enganar os resultados do modelo aluno. Primeiro é utilizado a perda L_2 para forçar o resultado do modelo aluno se aproximar do modelo professor pré-treinado. Para combater os métodos de detecção, o modelo aluno é treinado contra um discriminador que aprendeu a detectar amostras geradas e amostras falsas do Celeb-DF. Segundo ([PENG et al., 2021](#)), a função de perda do gerador é dada por

$$L_G = L_{BCE}(D(Y'), 0) + L_2(Y', Y), \quad (2.18)$$

onde o rótulo 0 na perda BCE representa o rótulo verdadeiro, que representa o resultado da geração do modelo aluno $D(Y')$ que precisa enganar o discriminador.

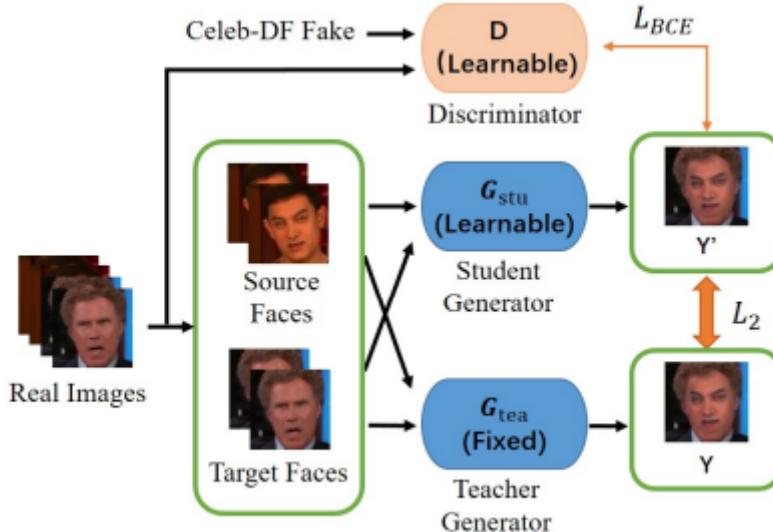
Para o treinamento adversarial, o discriminador aprende a classificar amostras X_{real} e X_{false} no conjunto de dados de treinamento do Celeb-DF, e também os resultados de troca de face do aluno Y' . Segundo (PENG et al., 2021), a função de perda do discriminador é dada por

$$L_D = L_{BCE}(D(Y'), 1) + L_{BCE}(D(X_{false}), 1) + L_{BCE}(D(X_{real}), 0), \quad (2.19)$$

onde o rótulo 0 representa real e 1 representa falso. O modelo aluno aprende a esconder as pistas de falsificação, resultando em uma melhor capacidade de não ser detectado.

Para as imagens terem uma melhor qualidade durante a criação de troca das faces, foi realizado uma seleção das imagens fonte. Foram selecionadas as faces que possuíam uma pose semelhante às faces alvo para gerar os resultados de troca de face. De acordo com (PENG et al., 2021), isso pode minimizar artefatos e distorções da face causadas por uma incompatibilidade da pose. A figura 13 mostra os passos que foram seguidos pelo trabalho.

Figura 13 – Passos seguidos pelo segundo colocado na geração de DeepFake.



Fonte: (PENG et al., 2021).

2.4.2 Detecção de DeepFake

2.4.2.1 Primeiro colocado

De acordo com (PENG et al., 2021), primeiro foi utilizado um detector de face MTCNN (ZHANG et al., 2016), o qual é utilizado para recortar as imagens da face de cada quadro do vídeo. Em seguida, foi utilizada a rede neural EfficientNet-B3 para realizar a detecção. E para

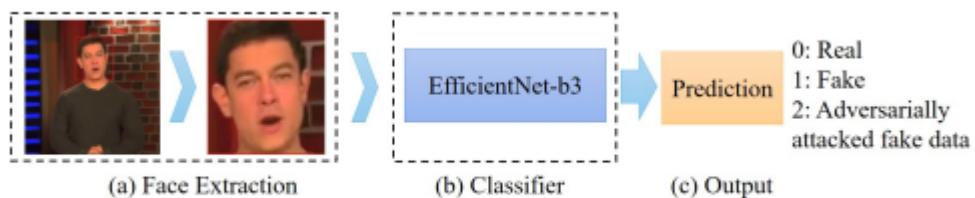
finalizar, a probabilidade de face real é obtida. Podemos ver a estrutura do detector de *DeepFake* na Figura 14.

O método *Face X-ray* (LI et al., 2020) consegue detectar artefatos de manipulação e de mesclagem. Com base nesse método o grupo que realizou esse trabalho utiliza um método semelhante para gerar mais imagens da face. Entretanto existe uma diferença, onde a imagem *DeepFake* é usada no primeiro plano e a imagem real é usada como fundo. Isso é feito pois uma é a região da face da imagem do *DeepFake* que contém artefatos de manipulação e a outra é o número de imagens *DeepFake* no Celeb-DF, que possui uma quantidade de imagens reais muito maior.

Para fazer com que a rede aprenda mais características, é empregado um método de aprendizado multiclasse com três neurônios de saída. Onde o primeiro p_0 denota a probabilidade da imagem real, o segundo p_1 denota a probabilidade da imagem falsa e o terceiro p_2 denota a probabilidade de dados falsos atacados por adversários. E a probabilidade final de um rosto falso durante o teste é calculada por $1 - p_0(x)$ ou $\sum_{i=1}^2 p_i(x)$.

Segundo (PENG et al., 2021), esse trabalho emprega um classificador baseado na arquitetura EfficientNet-B3 com uma perda de entropia cruzada de três classes. A técnica de suavização de rótulos é utilizada para evitar que o modelo preveja os rótulos com muita confiança durante o treinamento para melhorar a capacidade de generalização. As imagens são redimensionadas para 300×300 , o *batch size* é 8, a quantidade de épocas é 85 e a taxa de aprendizado é reduzida em 10% a cada 2 épocas.

Figura 14 – Estrutura do detector de *DeepFake* do primeiro colocado.



Fonte: (PENG et al., 2021).

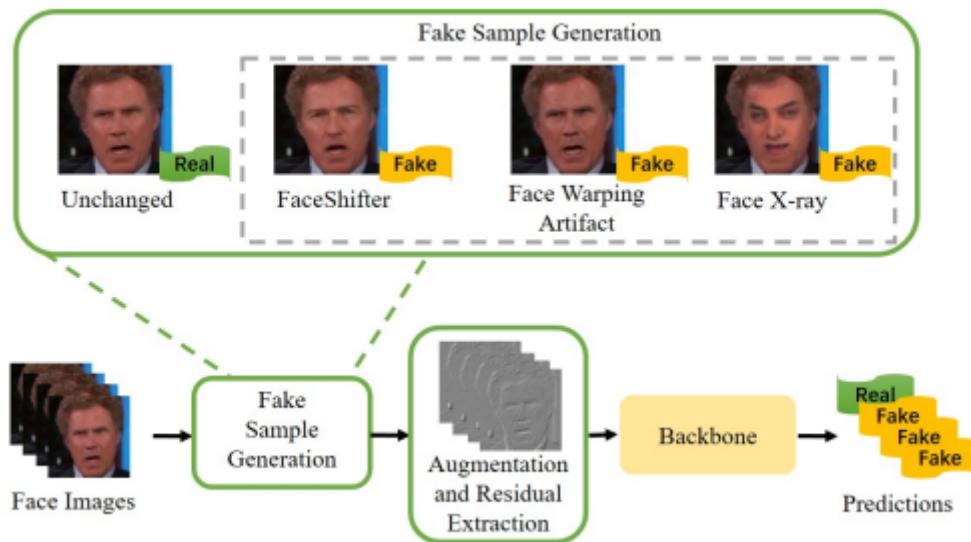
2.4.2.2 Segundo colocado

De acordo com (PENG et al., 2021), o segundo colocado apresenta um método de detecção que possui quatro estágios: pré-processamento, geração de amostras falsas, aumento de dados e um classificador baseado em CNN. É possível ver a estrutura do detector de *DeepFake* na Figura 15.

Para o pré-processamento é feito com uma amostra de 50 frames uniforme de cada vídeo. As faces são detectadas pelo MTCNN (ZHANG et al., 2016) e é feito um recorte conserva-

dor ao redor das faces detectadas. As faces recortadas são redimensionadas para 224×224 . Para gerar as amostras falsas são utilizados três métodos de geração de amostras que são: *FaceShifter*, *Face Warping Artifacts* e *Face X-ray*. Para o aumento de dados esse trabalho aplica vários métodos tradicionais de aumento de dados, onde eles citam transformação afim, compactação de imagem, desfoque gaussiano, dentre outros. A arquitetura utilizada foi a Efficientnet-b0, onde em vez da entrada das imagens RGB serem diretas, é extraído a borda das faces de entrada para obter recursos de alta frequência. No processo de treinamento, o *batch size* é definido como 64, a taxa de aprendizado é definida como 0.0005 utilizando o otimizador Adam e, em seguida, decai em 0,1 quando alcança as 5.000 iterações.

Figura 15 – Estrutura do detector de *DeepFake* do segundo colocado.



Fonte: ([PENG et al., 2021](#)).

3 Proposta do Trabalho

Neste trabalho foram feitos 2 métodos para realizar a detecção de *DeepFake*. O método 1 utiliza de uma CNN, para realizar a aprendizagem das imagens (reais e falsas) e classifica-las. Já no método 2 são utilizados métodos de qualidade de imagem para realizar a classificação das imagens reais e falsas.

3.1 Método 1

O método 1 possui 7 etapas que são representadas pelo diagrama mostrado na Figura 16. Esse método realiza o treinamento de uma CNN para detectar as imagens *DeepFakes*. Dentro do método 1 foram realizados dois experimentos, onde no primeiro experimento foram utilizadas imagens sem PRNU no treinamento da CNN e no segundo foram utilizadas imagens PRNU para o treinamento da CNN.

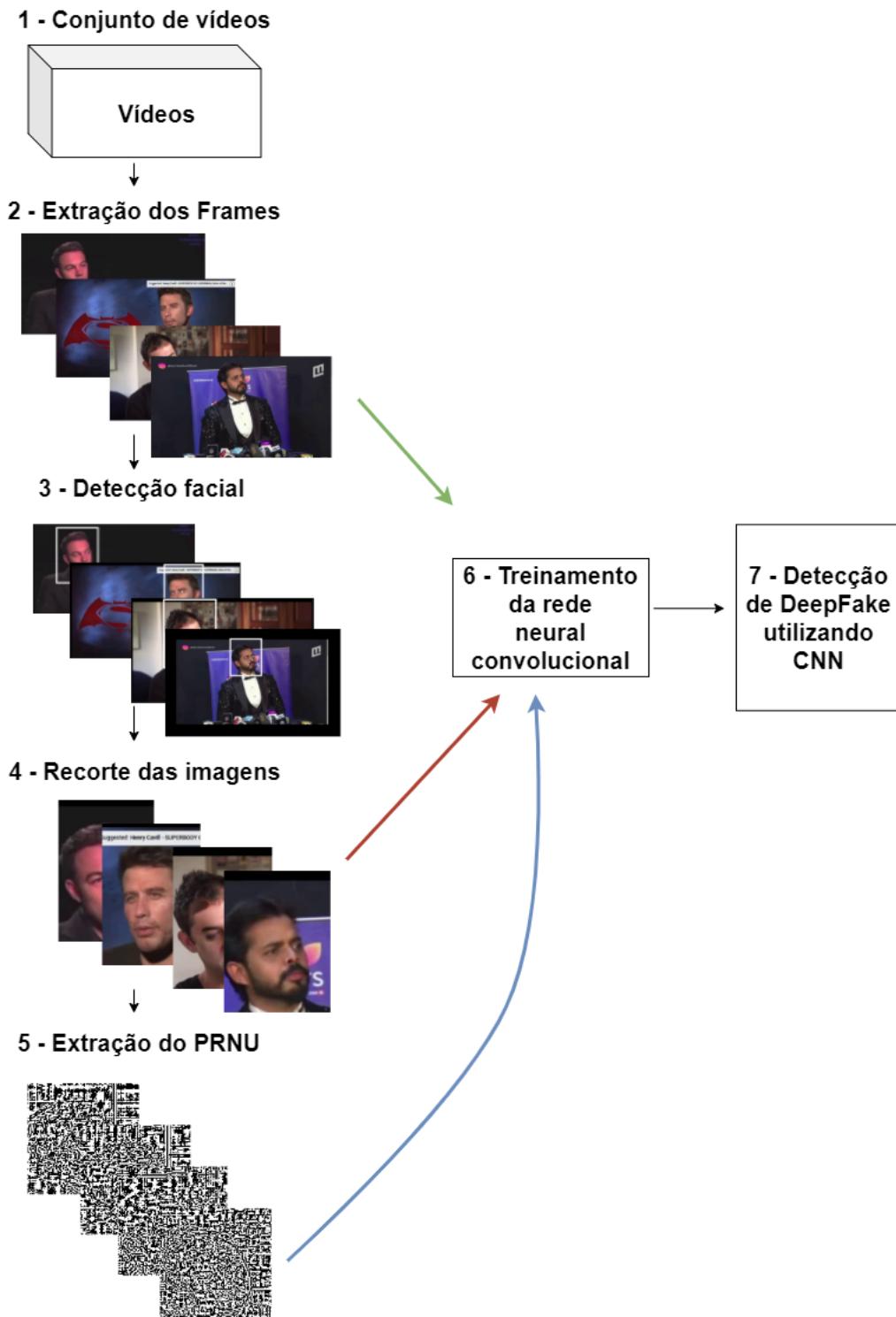
As etapas 1 e 2 desse diagrama representam a obtenção da base de imagens; a etapa 3 a detecção facial para posteriormente se fazer um recorte; a etapa 4 os recortes realizados em torno da face; a etapa 5 a extração do PRNU realizado de cada imagem; a etapa 6 o aprendizado realizado pelo treinamento da CNN (as 3 setas que se ligam a essa etapa, mostram as imagens que foram utilizadas para o treinamento da CNN); e para finalizar a etapa 7 representa a detecção de novas imagens *DeepFake*. Para melhor entendimento os próximos tópicos explicaram separadamente cada etapa.

3.1.1 Aquisição da Base de imagens

Nesta parte inicial foi pesquisado bases de imagens de *DeepFake* existentes na web. Durante a pesquisa foi encontrado o *dataset* disponibilizado pela competição DFGC 2021 ([PENG et al., 2021](#)). O DFGC 2021 foi uma competição que aconteceu na china no ano de 2021. Nesta competição os participantes precisavam criar uma gerador de *DeepFakes* e um detector de *DeepFakes*. Após a finalização da competição, as imagens criadas pelos participantes foram disponibilizadas para que a comunidade pudesse realizar mais pesquisas sobre a detecção de *DeepFake*.

Todas as imagens utilizadas neste trabalho foram disponibilizadas pelo DFGC. Nessa base de dados existem 18 classes, as quais 17 são de imagens *DeepFake* e 1 é de imagens reais. Cada classe possui mil imagens, dando um total de 18 mil imagens em todo a base de imagens. Para realizar o treinamento da CNN as imagens foram divididas em dois grupos (treino e teste), onde existem 601 imagens para treino e 399 para teste em cada classe. Dessa maneira, 10818 imagens foram utilizadas para treino e 7182 foram utilizadas para teste.

Figura 16 – Diagrama representando os passos a serem seguidos para a detecção de *DeepFakes* utilizando CNN.



Fonte: O autor.

3.1.2 Detecção Facial

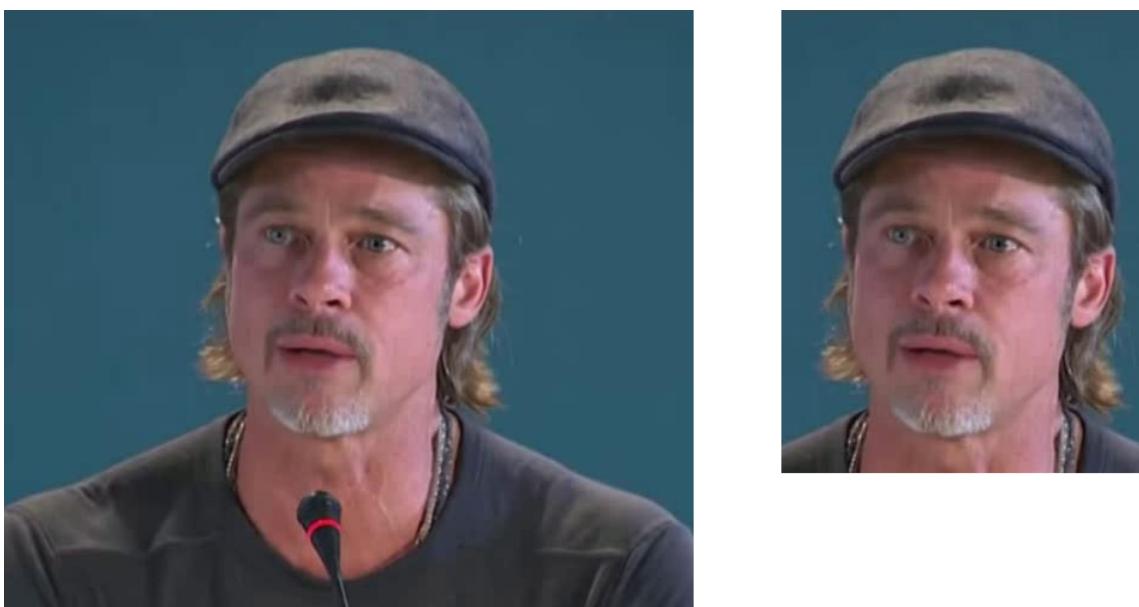
Queríamos realizar um recorte na imagem onde estava a pessoa. Para isso foi necessário utilizar o algoritmo MTCNN ([ZHANG et al., 2016](#)), para analisar cada imagem e fazer a detecção da face. A detecção foi feita tendo como pensamento principal que precisaríamos da face e

de parte do fundo da cena. Dessa maneira a área de recorte foi aumentada para que o fundo da imagem aparecesse junto com a face.

3.1.3 Recorte das imagens

Com a detecção facial realizado e a área aumentada, foi possível realizar o recorte na imagem. O recorte foi feito em cada uma das 18 mil imagens. O aumento da área de recorte foi necessário para que ainda existisse parte da imagem real dentro dessa nova imagem. Um exemplo do recorte pode ser visto na Figura 17.

Figura 17 – Exemplo de recorte realizado na face.



Fonte: O autor.

Dessa maneira, agora existem 2 bases de imagens que serão utilizadas futuramente nos experimentos.

- *Dataset 1:* Imagens originais.
- *Dataset 2:* Imagens com recorte na face.

3.1.4 Extração do PRNU

Foi utilizado o trabalho "Open Set Source Camera Attribution", realizado por ([COSTA et al., 2012](#)), como base para extraímos o PRNU de nossas imagens. O *dataset 2* foi utilizado para realizar a extração do PRNU ([COSTA et al., 2012](#)) de cada imagem. Para o PRNU ser criado, foi necessário calcular o ruído residual de cada imagem utilizando um filtro baseado no método da transformada de wavelet discreta, igual foi mostrado na equação [2.5](#). Contudo, foi feito 2 casos diferentes para essa extração de PRNU. No primeiro caso foi feito um *resize* na

imagem e no segundo caso não foi feito nenhuma alteração. Em seguida, foi utilizado um método de aprimoramento do padrão de ruído (normalização) para reduzir a influência do conteúdo da cena.

3.1.5 Treinamento da Rede Neural Convolucional

Nesse ponto existem as imagens dos *datasets* 1 e 2 e as imagens PRNU (com e sem *resize*) que foram extraídas do *dataset* 2. Dessa maneira, já é possível realizar o treinamento do algoritmo de CNN. Como podemos ver na Figura 16, existem três setas coloridas que ligam as imagens à CNN. A imagem está dessa maneira pois rodamos a CNN com diversas bases de imagens para verificar qual opção ofereceria um melhor resultado. A arquitetura utilizada no aprendizado da CNN foi a Resnet18.

3.1.6 Experimentos 1 e 2

Dois experimentos foram realizados no método 1. Para o experimento 1 foram utilizados os *datasets* 1 e 2 (sem extração do PRNU) para realizar o treinamento da CNN. Na Figura 16 isso é representado pela seta verde e vermelha.

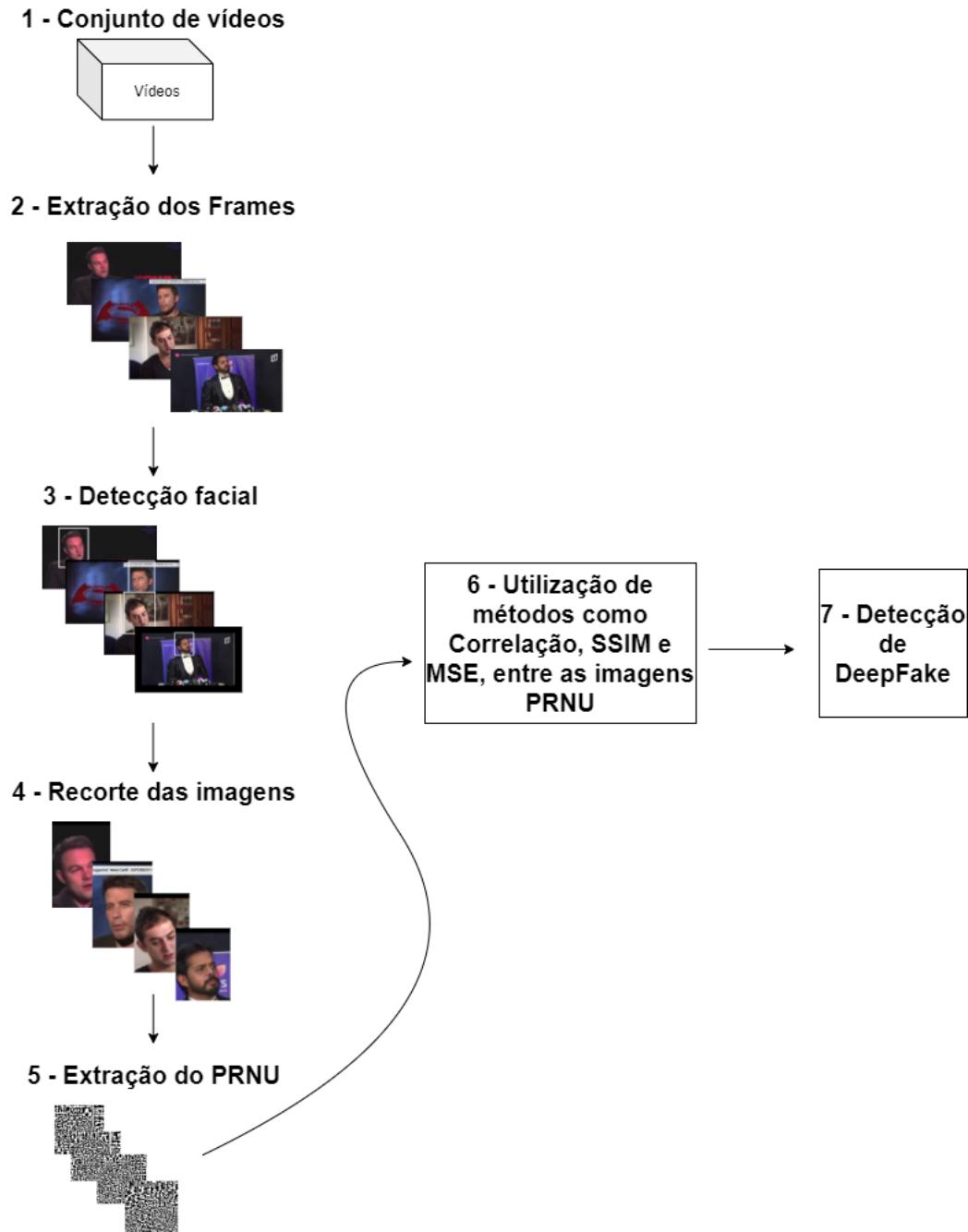
Em seguida foi realizado o experimento 2, onde apenas as imagens do *dataset* 2 foram utilizadas. Neste experimento foram realizados dois testes utilizando em um dos testes um *resize* nas imagens. No primeiro teste as imagens do *dataset* 2 sofreram um *resize* e no segundo elas não sofreram nenhuma alteração. Agora que temos imagens com e sem *resize*, foi realizado a extração do PRNU de cada imagem. Para finalizar, a CNN foi treinada para cada um dos dois testes. Isso pode ser visto na Figura 16 pela seta azul.

3.2 Método 2

O método 2 possui novamente 7 etapas que são representadas pelo diagrama mostrado na Figura 18. Esse método utiliza de métodos de qualidade de imagem (correlação, MSE e SSIM) para realizar a detecção de imagens *DeepFakes*. Dentro do método 2 foi realizado o terceiro experimento.

As etapas 1, 2, 3 e 4 desse diagrama efetuam os mesmos procedimentos realizados no método 1, dessa maneira elas não serão repetidas. A etapa 5 representa extração do PRNU realizado de cada imagem, porém dessa vez não foi realizado *resize* e nem normalização; a etapa 6 representa a utilização de métodos de qualidade de imagem para criar matrizes; e para finalizar a etapa 7 representa a detecção de novas imagens *DeepFake* através da métrica de acurácia top-*N*.

Figura 18 – Diagrama representando os passos a serem seguidos para a detecção de *DeepFakes* utilizando métodos de medida de qualidade da imagem.



Fonte: O autor.

3.2.1 Extração do PRNU

Dessa vez novamente foi utilizado o trabalho "Open Set Source Camera Attribution", realizado por ([COSTA et al., 2012](#)), como base para extraímos o PRNU de nossas imagens. Apenas o *dataset* 2 foi utilizado para realizar a extração do PRNU ([COSTA et al., 2012](#)) de cada imagem, entretanto não foram utilizados *resize* e nem normalização nas imagens. Dessa vez extraímos o PRNU das imagens e realizamos um corte em seu centro. De acordo com

(COSTA et al., 2012), o melhor local para realizar esse corte na imagem é no centro.

3.2.2 Utilização de Medidas de Qualidade da Imagem

Agora que temos as imagens PRNU com recorte no meio, foi feito uma verificação de semelhança entre os PRNUs. Para realizar a verificação da semelhança entre as imagens foram utilizados os métodos de correlação, SSIM e MSE. Cada imagem do conjunto de teste foi verificada com todas as imagens do conjunto de treino. Dessa maneira foram geradas 3 matrizes, de tamanho 7182×10818 , com os resultados de cada um dos métodos.

3.2.3 Utilização da métrica Acurácia Top-N

Para realizar a detecção de *DeepFake* dessa vez alguns passos foram seguidos. Temos que ter em mente que cada linha das matrizes geradas no passo anterior contém o valor do coeficiente de semelhança entre uma imagem de teste e todas as imagens de treino. Para utilizar a acurácia top- N , é necessário que as imagens de treino mais parecidas com as imagens testem estejam no começo da linha. Dessa maneira, é necessário realizar a ordenação dos valores com suas respectivas classes de treino. Para os métodos correlação e SSIM foi feito uma ordenação decrescente, e para o método MSE foi feito uma ordenação crescente. Após a ordenação é realizada uma acurácia top- N entre as primeiras N imagens de treino. Isso é, verificar se existe pelo menos uma aparição da classe de teste dentro das primeiras N imagens de treino.

3.2.4 Experimento 3

Apenas o *dataset* 2 foi utilizado para realizar o experimento 3. Para começar foi feito um recorte central de tamanho 50×50 em todas as imagens. Após, foi retirado o PRNU de cada imagem sem realizar a normalização. Em seguida, foi utilizado os métodos de qualidade de imagem (correlação, SSIM e MSE) para realizar a verificação da semelhança entre cada imagem de teste com todas as imagens de treino. Dessa maneira, foram geradas 3 matrizes possuindo os coeficientes de similaridade dos três métodos de qualidade de imagem. Os valores contidos nas matrizes foram ordenados e para finalizar, foi utilizado a métrica de acurácia top- N para realizar a detecção de *DeepFakes*.

4 Resultados e discussão

4.1 Metodologia Experimental

4.1.1 Base de Imagens

Para criar nossa base de imagens utilizamos o *dataset* disponibilizado pelo DFGC 2021. De acordo com (PENG et al., 2021), o DFGC é parcialmente inspirada por duas competições recentes realizadas na China, que avaliam os lados da criação e detecção de falsificações. Dessa maneira, a competição é dividida em duas etapas: a criação do *DeepFake* e detecção do *DeepFake*, e as etapas são avaliadas uma contra a outra. Os organizadores do DFGC são a Chinese Academy of Sciences, a Ocean University of China e a University at Buffalo, e o patrocinador é o Tianjin Academy for Intelligent Recognition Technologies.

Segundo (PENG et al., 2021), para uma avaliação justa e uma base comum para o treinamento do modelo, é exigido que a criação e detecção sejam baseadas em um conjunto de dados restrito, para o qual foi escolhido o conjunto de dados Celeb-DF v2. Esse conjunto de dados é composto por 590 vídeos reais e 5.639 vídeos falsos de alta qualidade que são criados por um AutoEncoder modificado baseado em método de *DeepFake*. Para a criação das imagens falsas, foi especificado 1.000 imagens de *DeepFakes* a serem criadas e enviadas pelos participantes. Essas imagens são especificadas a partir dos vídeos falsos do conjunto de testes Celeb-DF.

Dessa maneira pegamos o *dataset* fornecido pelo DFGC que possui 18 classes, sendo elas: 17 classes de *DeepFakes* diferentes e uma classe com imagens reais. A base de imagens possui 1000 imagens em cada classe, dando um total de 18 mil imagens em toda a base. Dessas imagens foram utilizadas 601 imagens para treino e 399 para teste em cada classe, dando um total de 10818 e 7182 para treino e teste respectivamente. Podemos ver um exemplo de cada GAN e das imagens reais na Figura 19.

A partir do *dataset* fornecida pelo DFGC, geramos mais uma base realizando o recorte sobre a base de imagens. Dessa maneira ficamos com 2 *datasets* que são:

- *Dataset 1*: imagens originais do DFGC sem PRNU;
- *Dataset 2*: imagens com um recorte realizado perto da face sem PRNU;

Onde cada *dataset* possui 18 mil imagens. Podemos ver um exemplo de cada base na Figura 20.

Para criarmos o *dataset 2* precisamos utilizar o algoritmo MTCNN (ZHANG et al., 2016), explicado no trabalho "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks" feito por (ZHANG et al., 2016), para realizar os recortes nas imagens.

Figura 19 – Recortes realizados na face.



Fonte: O autor.

O algoritmo já identifica a face em cada imagem, criando uma área retangular em volta da face. Dessa maneira foi necessário apenas aumentar o tamanho dessa área retangular. Para realizar o aumento dobramos a distância do centro até a extremidade em cada vértice do retângulo.

4.1.2 PRNU utilizado na CNN

Antes de realizar a extração do PRNU temos dois casos. No primeiro é feito um resize com as dimensões de 128×128 , no segundo nenhuma alteração é feita na imagem. Em seguida foi feita a extração do PRNU utilizando um filtro baseado no método da transformada de wavelet discreta e uma normalização após a utilização do filtro. Os valores utilizados em α na

Figura 20 – Exemplo de datasets utilizados.



Fonte: O autor.

normalização foi de 1, 7, 10, 20. Podemos ver dois exemplos de extração de PRNU na Figura 21.

Figura 21 – Exemplo de extração de PRNU.



Fonte: O autor.

4.1.3 Rede Neural Convolucional

Foi utilizado a arquitetura Resnet18 e a biblioteca pytorch para realizar o treinamento da Rede Neural Convolucional. Foram considerados os seguintes parâmetros no algoritmo:

- transforms.Compose: CenterCrop = 32; ToTensor; Normalize = ((0.5, 0.5, 0.5), (0.5, 0.5, 0.5));
- DataLoader: *batch size* = 128; shuffle = True para treino e shuffle = False para teste; num_workers = 2;
- Função de perda: nn.CrossEntropyLoss;
- Função de otimização: optim.SGD;
- Função de definição da taxa de aprendizado: optim.lr_scheduler.CosineAnnealingLR
- Epochs: valor igual a 200;

4.1.4 PRNU utilizado nos métodos de medida de qualidade da imagem

Foi realizada a extração do PRNU utilizando um filtro baseado no método da transformada de wavelet discreta. Após a extração do PRNU, foi feito um corte no centro da imagem de dimensão 50×50

4.1.5 Métodos de medida de qualidade da imagem

Os métodos utilizados foram a correlação, SSIM e MSE. Assim que utilizamos esses métodos foram obtidas 3 matrizes com tamanho 7182×10818 , onde cada linha representa as imagens de teste e as colunas representam as imagens de treino. Dessa maneira, era necessário utilizar uma ordenação em cada linha da matriz. Para os métodos correlação e SSIM foi feito uma ordenação dos valores decrescente, já para o método MSE foi feito uma ordenação crescente. As classes das imagens de treino tiveram que ir junto com os valores em forma de tupla, para não se perder a classe original daquela semelhança. Após termos todos os valores ordenados com suas devidas classes, foi decidido uma quantidade N de imagens para realizar a acurácia top- N . Os valores para N foram 1, 2, 3, 4, 5 e 10.

4.2 Experimentos e Discussões

4.2.1 Experimento 1: classificação das imagens sem PRNU utilizando CNN

Os resultados da classificação utilizando CNN utilizando as imagens sem PRNU estão na tabela 4. O melhor resultado possui 56.7% de acurácia. Ele ocorre quando utilizamos o *dataset* 2, ou seja, quando fazemos um recorte um pouco maior que a face existente na imagem.

Tabela 4 – Resultados dos *datasets* 1 e 2 sem a utilização de PRNU.

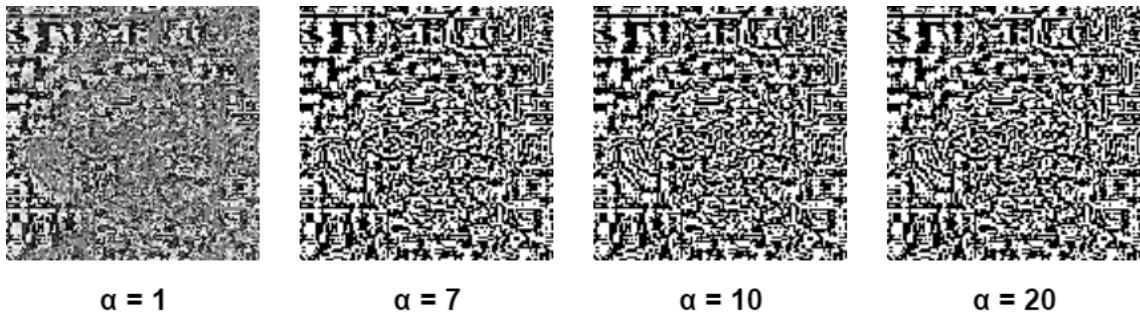
Dataset	Acurácia
1	24.79%
2	70.8%

Fonte: O Autor

4.2.2 Experimento 2: classificação das imagens com PRNU utilizando CNN

Vamos ter 2 casos de classificação nesse experimento, ambos foram realizados com o *dataset* 2. Primeiro será mostrado o caso onde é realizado um resize na imagem e a extração do PRNU. Podemos ver os resultados na tabela 5. A maior acurácia nesse caso é de 46.1% quando temos o $\alpha = 7$. Podemos ver um exemplo das imagens geradas com os 4 diferentes alphas na Figura 22.

Figura 22 – Exemplo das imagens com os 4 alphas utilizados.



Fonte: O autor.

Tabela 5 – Resultados do *dataset* 2 utilização PRNU e resize.

Alpha	Acurácia
1	37.28%
7	46.10%
10	40.26%
20	34.61%

Fonte: O Autor

O segundo caso é quando temos apenas a extração do PRNU. Podemos ver os resultados na tabela 6. A maior acurácia obtida nesse caso é de 32.16% quando temos $\alpha = 7$.

Tabela 6 – Resultados do *dataset* 2 utilização apenas PRNU.

Alpha	Acurácia
1	31.41%
7	32.16%
10	28.19%
20	25.49%

Fonte: O Autor

No experimento 2 o melhor resultado foi de acurácia igual a 46.1%, quando utilizamos as imagens com resize, extração de PRNU e com α igual a 7.

4.2.3 Experimento 3: classificação das imagens PRNU utilizando acurácia top-N

A Tabela 7 mostra os resultados de classificação utilizando a medida de acurácia top- N . Como é possível observar, a medida de qualidade SSIM consegue pequena melhora na aplicação alvo quando comparados a outras duas medidas (correlação e MSE).

Tabela 7 – Acurácia top-*N* dos métodos Correlação, SSIM e MSE utilizando PRNU.

Top-N	Correlação	SSIM	MSE
1	12%	15%	6%
2	21%	24%	10%
3	27%	32%	12%
4	33%	38%	14%
5	38%	44%	16%
10	58%	64%	20%

Fonte: O Autor

4.2.4 Discussão geral

O *DeepFake* é um assunto muito importante a se discutir atualmente. A falta de conteúdos que ajude na detecção pode vir a se tornar um problema daqui a um tempo. Com isso em mente, esse trabalho procurou achar maneiras de vir a tratar o problema da detecção de *DeepFakes*. No começo do trabalho, foi visto o trabalho "Open Set Source Camera Attribution" de (COSTA et al., 2012), que tratava sobre uma maneira de encontrar qual câmera teria tirado uma foto específica. A técnica de PRNU é utilizada, na ciência forense, para se encontrar possíveis câmeras que teriam tirado alguma foto criminosa. Tendo isso em mente, foi pensado se o PRNU também poderia encontrar a GAN que teria criado um *DeepFake*. Dessa maneira o trabalho foi iniciado, buscando uma maneira de utilizar o PRNU para detectar *DeepFake*.

Neste trabalho foram feitos três testes distintos. No experimento 1, o melhor resultado obtido foi de 56.7% de acurácia utilizando o *dataset* 2. Onde uma CNN foi treinada com 18 classes distintas com imagens sem nenhuma alteração. Dentro desse experimento já era esperado que o melhor resultado seria o das imagens recortadas. Em outros trabalhos de detecção de *DeepFake*, também é utilizado o algoritmo MTCNN (ZHANG et al., 2016) para fazer um recorte em torno da face, como foi falado na seção 2.4. Com isso foi possível comprovar que realmente é necessário realizar um recorte na face para encontrar melhores resultados na detecção de *DeepFake*.

No experimento 2 o melhor resultado obtido foi de 46.10% de acurácia também utilizando o *dataset* 2. Nesse teste, uma CNN foi treinada com 18 classes distintas. As imagens receberam dois pré-processamentos, o primeiro com um recorte na face, utilizando o algoritmo MTCNN (ZHANG et al., 2016), e o segundo fazendo extração do PRNU. Como já conseguimos verificar que a melhor maneira de detectar *DeepFake* é utilizando apenas as imagens com recorte, neste teste não foram utilizando as imagens originais (*dataset* 1). Agora focando na extração do PRNU, pode-se perceber que o melhor α era o de valor 7. Esse valor também foi o citado no trabalho de (COSTA et al., 2012), entretanto não era esperado que o valor fosse o mesmo. Realizando diversos testes foi possível perceber que conforme o valor de α se afasta de 7, o valor de acurácia da CNN vai diminuindo. Contudo, consideramos um problema encontrar um valor de α para o PRNU normalizado. Foi percebido que os PRNUs não estavam

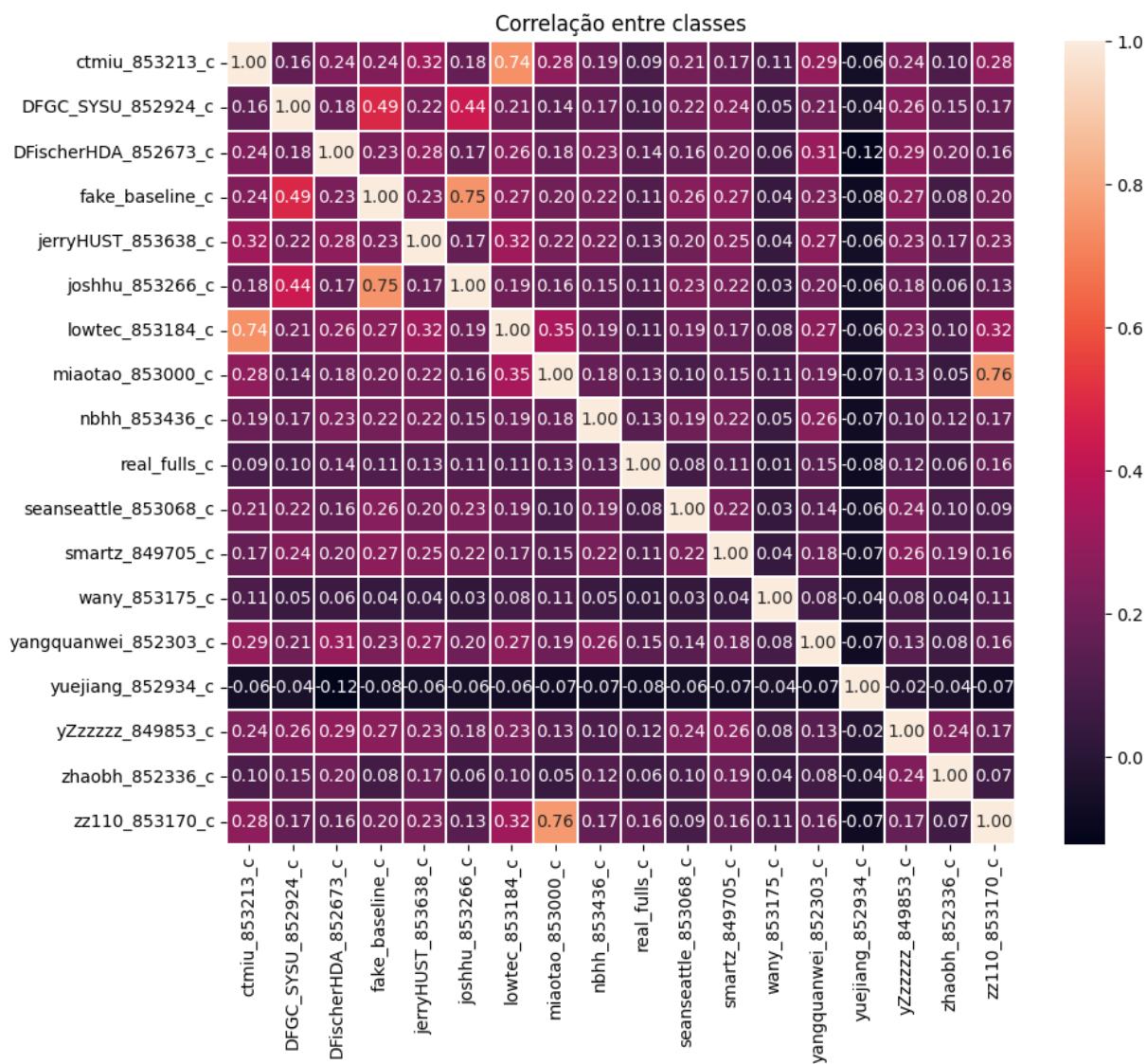
representando as imagens reais e fake.

E para finalizar, no experimento 3 o melhor resultado obtido foi de 64%. Entretanto nesse teste não é mais utilizado uma CNN. Novamente nesse experimento é utilizado apenas as imagens do *dataset 2*, onde elas recebem um recorte central de 50×50 pixels e é extraído os seus respectivos PRNUs. Dessa vez não é realizado a normalização no PRNU, pois notamos que a normalização estava confundindo a detecção. Com as imagens já processadas, é feito uma verificação de similaridade entre as imagens utilizando os métodos de qualidade de imagem, e com os seus resultados, é feito uma classificação utilizando o método de Acurácia Top-*N*. Esse experimento foi proposto, pois surgiu uma dúvida se seria possível utilizar uma abordagem sem CNN. E como podemos notar realmente foi possível. Nesse experimento foi possível analisar que o método SSIM consegue realizar uma comparação mais fiel, do que os outros dois métodos. Outro ponto a se comentar, é na utilização da acurácia Top-10 como melhor resultado. Apesar de pelo menos uma imagem ser da classe certa dentro de 10 imagens, temos que notar que existem 10818 possibilidades que poderiam estar nas primeiras 10 posições. Com esse pensamento, conseguimos chegar à conclusão que utilizar uma Acurácia Top-10 não é ruim.

Ainda no experimento 3, podemos verificar a imagem 23, onde temos uma matriz de correlação entre as classes. Ela foi feita com a imagem média gerada pelas imagens PRNUs contidas em cada classe, ou seja, todas as imagens de cada classe foram somadas e divididas pela quantidade de imagens da classe (1000 imagens). Para finalizar foi feita a correlação entre todas as imagens PRNU médias. Analisando a Figura 23, podemos verificar que a correlação entre as imagens PRNU médias são baixas entre si, ou seja, elas são diferentes umas das outras. Com isso, pensamos que poderíamos ter bons resultados utilizando correlação, porém ao plotar o TSNE (MAATEN; HINTON, 2008) verificamos que esse dado era enganoso. Olhando para a Figura 24, podemos observar o TSNE feito com as imagens PRNU, porém todas estão misturadas nessa representação. Isso indica que ao realizar a extração do PRNU, possivelmente surgiu uma semelhança de comportamento entre as classes, o que confundiu os classificadores. Podemos verificar melhor a mistura de classes na Figura 25, onde foi feito uma aproximação (*Zoom in*) no TSNE.

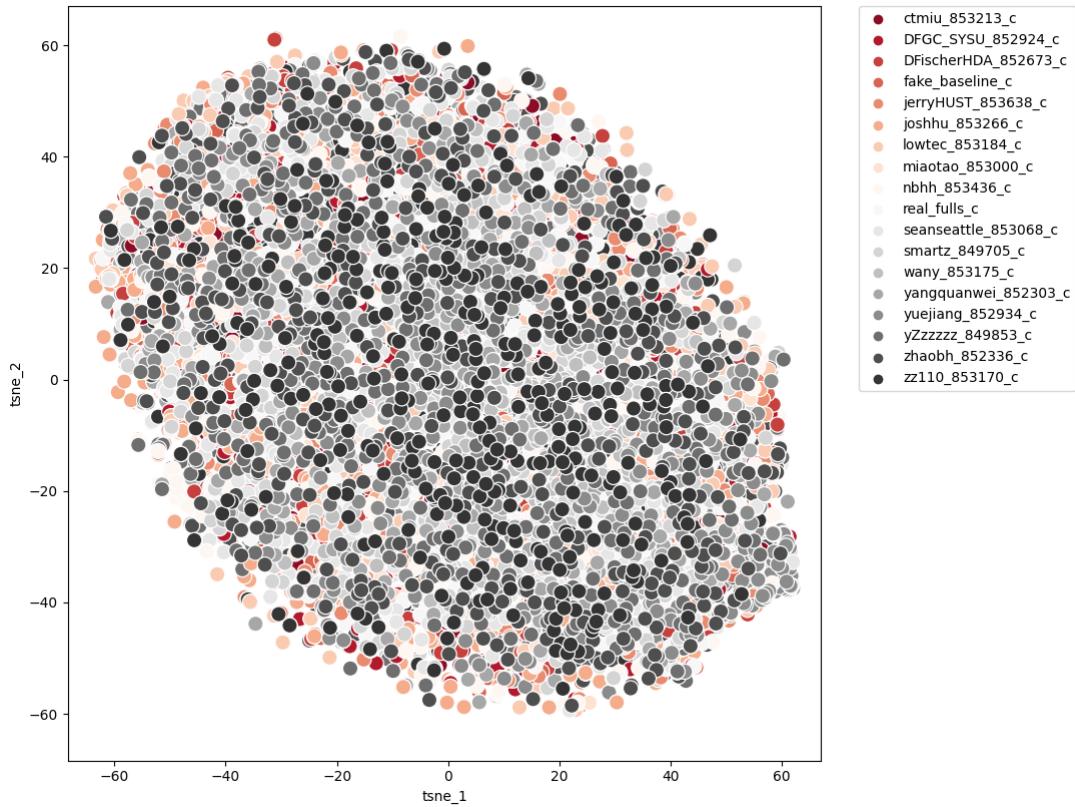
Com esses resultados obtidos foi possível alcançar o objetivo proposto para esse trabalho, que seria conseguir realizar a detecção de *DeepFakes* utilizando a técnica de PRNU. Entretanto esperávamos que os resultados obtidos através da CNN seriam maiores. Possivelmente a baixa acurácia dos testes realizados com as imagens PRNUs é devido a semelhança no comportamento entre as classes.

Figura 23 – Mapa de calor da correlação entre as classes.



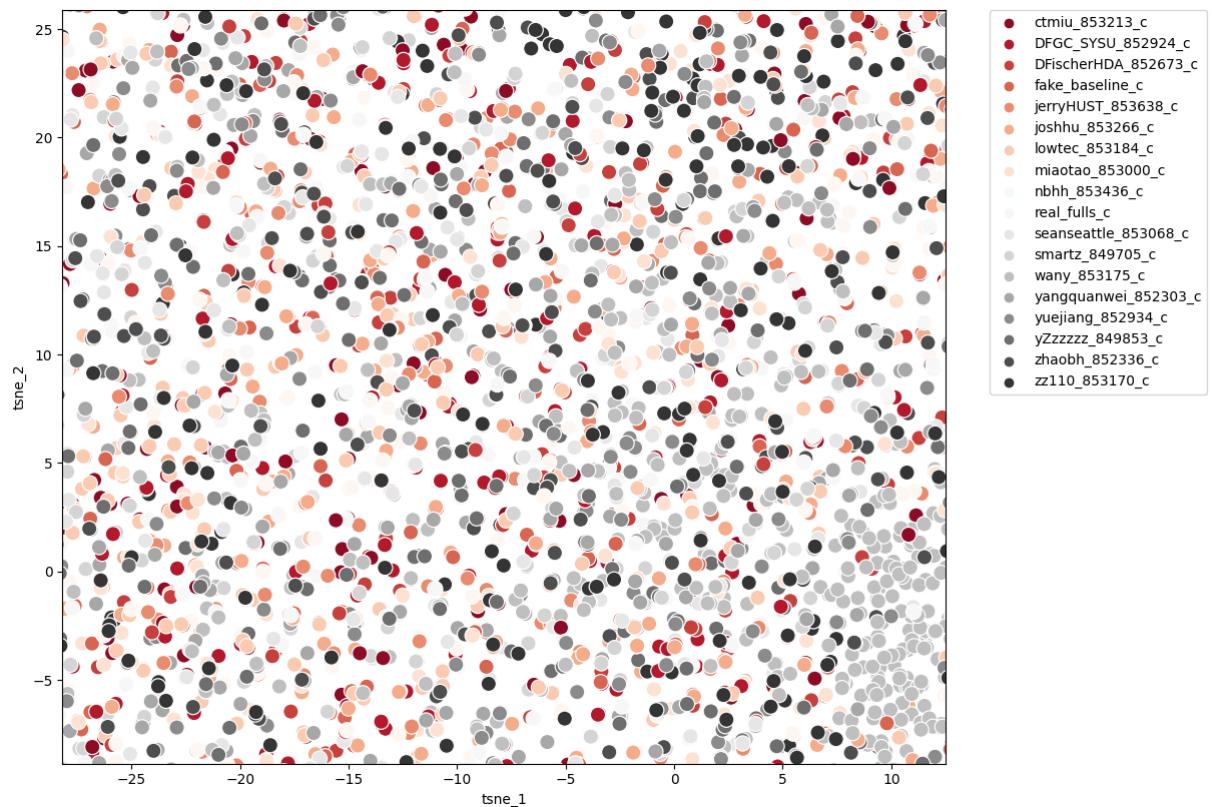
Fonte: O autor.

Figura 24 – TSNE das imagens PRNU com suas respectivas classes.



Fonte: O autor.

Figura 25 – Aproximação do TSNE das imagens PRNU.



Fonte: O autor.

5 Conclusão

A manipulação de imagens e vídeos é não é uma tarefa recente, softwares como o Photoshop, conseguem realizar a tarefa de editar imagens, contudo é fácil reconhecer as imagens falsas feitas dessa maneira. Entretanto com o avanço na área de inteligência artificial, surgiram novas maneiras de alterar imagens, como por exemplo o *DeepFake* utilizando de aprendizagem profunda. O termo *DeepFake* se refere a geração de conteúdos multimídias muito realistas onde as pessoas não conseguem diferenciar entre real e falso. Essa técnica pode ser aplicada para situações construtivas, como entretenimento e educação, mas também pode ser utilizada para práticas antiéticas. Infelizmente foi as práticas ruins que deixaram o *DeepFake* famoso. Mais casos de usos maliciosos começaram a surgir desde a primeira aparição do método *DeepFake* em 2017 no *Reddit*. Tendo em vista que ela pode ser utilizada para o mal, se iniciou um movimento mundial de combate de tais práticas, iniciando uma pesquisa de métodos para detectar *DeepFake*.

Tendo em vista os problemas causados pelo *DeepFake*, este trabalho realiza o estudo e desenvolvimento de métodos para detectar *DeepFake*, que se dividi em três experimentos. No experimento 1, uma CNN é treinada com dois *datasets*. O *dataset* 1 possui as imagens sem nenhuma alteração e o *dataset* 2 possui as imagens com um recorte na face. O melhor resultado obtido nesse experimento foi uma acurácia de 56.7% utilizando o *dataset* 2. Com esse teste, conseguimos comprovar que existe uma melhora quando se faz um recorte em torno da face.

No experimento 2, foram realizados dois testes. No primeiro foi feito um recorte, *resize* e extração do PRNU normalizado, com diferentes valores para α , utilizando o *dataset* 2. E no segundo ocorreu quase o mesmo pré-processamento de imagens com o *dataset* 2, entretanto a imagem não recebeu um *resize*. Em seguida, a CNN foi treinada com as imagens PRNU dos dois testes. O melhor resultado obtido foi uma acurácia de 46.1%, ao utilizar *resize*, e $\alpha = 7$ na normalização do PRNU.

Para finalizar, no experimento 3 foi realizado a extração do PRNU sem normalização no *dataset* 2 e um recorte de dimensão 50×50 no centro das imagens geradas. Em seguida, foi criado uma matriz com os coeficientes das medidas de qualidade da imagem (correlação, SSIM e MSE), das imagens de teste com as imagens de treino. Por fim, foi utilizado o método acurácia top- N para a classificação das imagens. O melhor resultado obtido foi de uma acurácia top-10 de 64%, ao se utilizar o SSIM como método de qualidade de imagem.

Com esses resultados, conseguimos concluir que o método de acurácia top- N pode apresentar um resultado melhor que uma CNN. Também é possível afirmar que a utilização de imagens originais carrega mais informações relevantes para a tarefa alvo que as imagens resultantes da extração PRNU.

Para trabalhos futuros várias ações podem ser tomadas a partir desse ponto. Podemos citar como exemplo: a busca de novos métodos de normalização para as imagens resultantes do PRNU, já que um dos problemas deste trabalho foi encontrar um PRNU que representasse as imagens reais e falsas; o estudo e avaliação comparativa de arquiteturas de CNNs para o problema alvo; aplicar técnicas de aumento de dados para este problema, buscando uma melhora na eficácia dos resultados; realizar um estudo e análise em outros *datasets* existentes na literatura.

Referências

- AGARAP, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. Citado na página 36.
- ARNOLD, L. et al. *An introduction to deep learning*. In: *Advances in Computational Intelligence and Machine Learning*. [S.l.], 2011. 13 p. Citado na página 36.
- BIANCHINI, Â. R. Arquitetura de redes neurais para o reconhecimento facial baseado no neocognitron. Universidade Federal de São Carlos, 2001. Citado 3 vezes nas páginas 33, 34 e 35.
- BUSLAEV, A. et al. Albumentations: fast and flexible image augmentations. *Information*, Multidisciplinary Digital Publishing Institute, v. 11, n. 2, p. 125, 2020. Citado na página 44.
- COSTA, F. d. O. et al. Open set source camera attribution. In: IEEE. *2012 25th SIBGRAPI conference on graphics, patterns and images*. [S.l.], 2012. p. 71–78. Citado 6 vezes nas páginas 29, 32, 51, 53, 54 e 60.
- FRIED, O. et al. Text-based editing of talking-head video. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 38, n. 4, p. 1–14, 2019. Citado na página 44.
- GANDHI, S. A.; KULKARNI, C. Mse vs ssim. *International Journal of Scientific & Engineering Research*, v. 4, n. 7, p. 930–934, 2013. Citado 2 vezes nas páginas 30 e 31.
- GHOLAMALINEZHAD, H.; KHOSRAVI, H. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020. Citado na página 38.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 36.
- GOODFELLOW, I. et al. Generative adversarial nets. *Advances in neural information processing systems*, v. 27, 2014. Citado na página 39.
- HAYKIN, S. Redes neurais: princípios e prática, 2^a edição, tradução: Paulo martins engel. *Editora: Bookman, Porto Alegre, Cap*, v. 1, n. 2, p. 3, 2001. Citado 2 vezes nas páginas 32 e 33.
- HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969. Citado na página 43.
- JOST, I. Aplicação de deep learning em dados refinados para mineração de opiniões. Universidade do Vale do Rio dos Sinos, 2015. Citado na página 41.
- KARNOUSKOS, S. Artificial intelligence in digital media: The era of deepfakes. *IEEE Transactions on Technology and Society*, IEEE, v. 1, n. 3, p. 138–147, 2020. Citado na página 43.
- KIM, H. et al. Deep video portraits. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 37, n. 4, p. 1–14, 2018. Citado na página 44.

KOECH, K. E. Cross-entropy loss function. *Medium: towards data science*, 2020. Citado na página 35.

LI, L. et al. Face x-ray for more general face forgery detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2020. p. 5001–5010. Citado na página 47.

MAATEN, L. Van der; HINTON, G. Visualizing data using t-sne. *Journal of machine learning research*, v. 9, n. 11, 2008. Citado na página 61.

MIRSKY, Y.; LEE, W. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 54, n. 1, p. 1–41, 2021. Citado na página 23.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole, v. 1, n. 1, p. 32, 2003. Citado 2 vezes nas páginas 32 e 39.

NGUYEN, T. T. et al. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint arXiv:1909.11573*, 2019. Citado na página 43.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais: princípios, algoritmos e aplicações*. [S.l.]: Cengage Learning, 2008. Citado na página 27.

PENG, B. et al. Dfgc 2021: A deepfake game competition. In: *IEEE. 2021 IEEE International Joint Conference on Biometrics (IJCB)*. [S.l.], 2021. p. 1–8. Citado 8 vezes nas páginas 24, 44, 45, 46, 47, 48, 49 e 55.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. *Rita*, v. 13, n. 2, p. 11–42, 2006. Citado na página 27.

RODRIGUES, D. A. Deep learning e redes neurais convolucionais: reconhecimento automático de caracteres em placas de licenciamento automotivo. Universidade Federal da Paraíba, 2018. Citado 4 vezes nas páginas 34, 35, 36 e 37.

SANTANA, T. O. d. Comparação entre técnicas de aprendizado de máquina no processo de identificação biométrica através de imagens da orelha. 2019. Citado na página 38.

SANTOS, J. d. S. et al. Geração e aplicação de imagens sintéticas da região dos olhos para o aumento de performance de técnicas da detecção do estrabismo. Universidade Presbiteriana Mackenzie, 2020. Citado na página 39.

SILVA, A. V. da. *Transformada Wavelets - abordagem de sua aplicabilidade*. [S.l.], 2014. 15 p. Citado 3 vezes nas páginas 27, 28 e 29.

SUWAJANAKORN, S.; SEITZ, S. M.; KEMELMACHER-SHLIZERMAN, I. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (ToG)*, ACM New York, NY, USA, v. 36, n. 4, p. 1–13, 2017. Citado na página 44.

SWATHI, P.; SK, S. Deepfake creation and detection: A survey. In: *IEEE. 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*. [S.l.], 2021. p. 584–588. Citado 2 vezes nas páginas 23 e 24.

THIES, J. et al. Face2face: Real-time face capture and reenactment of rgb videos. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 2387–2395. Citado na página 44.

- VIEIRA, V. et al. Tópicos em gerenciamento de dados e informações: Minicursos do sbbd 2017. *Tópicos*, v. 2, p. 10, 2017. Citado 3 vezes nas páginas [35](#), [36](#) e [37](#).
- WANG, Z.; BOVIK, A. C. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, IEEE, v. 26, n. 1, p. 98–117, 2009. Citado na página [31](#).
- YU, P. et al. A survey on deepfake video detection. *IET Biometrics*, Wiley Online Library, v. 10, n. 6, p. 607–624, 2021. Citado 5 vezes nas páginas [23](#), [24](#), [42](#), [43](#) e [44](#).
- ZAID, M. A. Correlation and regression analysis. *The Statistical, Economic and Social Research and Training Centre for Islamic Countries (SESRIC)*, Turkey, 2015. Citado na página [31](#).
- ZHANG, K. et al. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, IEEE, v. 23, n. 10, p. 1499–1503, 2016. Citado 5 vezes nas páginas [46](#), [47](#), [50](#), [55](#) e [60](#).