Universidade Estadual de Campinas
Instituto de Computação

Camila Steffane Fernandes Teixeira de Moura

Detecção de *DeepFakes* a Partir de Técnicas de Visão Computacional e Aprendizado de Máquina

CAMPINAS
2021

Camila Steffane Fernandes Teixeira de Moura

Detecção de *DeepFakes* a Partir de Técnicas de Visão Computacional e Aprendizado de Máquina

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Ciência da Computação.

Orientador: Prof. Dr. Anderson de Rezende Rocha

Este exemplar corresponde à versão final da Dissertação defendida por Camila Steffane Fernandes Teixeira de Moura e orientada pelo Prof. Dr. Anderson de Rezende Rocha.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

M765d

Moura, Camila Steffane Fernandes Teixeira de, 1991-
Detecção de Deepfakes a partir de técnicas de visão computacional e aprendizado de máquina / Camila Steffane Fernandes Teixeira de Moura. – Campinas, SP : [s.n.], 2021.

Orientador: Anderson de Rezende Rocha.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Detecção de deepfakes. 2. Aprendizado de máquina. 3. Visão por computador. 4. Imagens manipuladas. I. Rocha, Anderson de Rezende, 1980-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: DeepFake detection through computer vision and machine learning techniques
Palavras-chave em inglês:
Deepfakes detection
Machine learning
Computer vision
Manipulated images
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
Anderson de Rezende Rocha [Orientador]
João Paulo Papa
Julio Cesar dos Reis
Data de defesa: 01-03-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-3455-1209
- Currículo Lattes do autor: http://lattes.cnpq.br/9515478432955890

Universidade Estadual de Campinas
Instituto de Computação

Camila Steffane Fernandes Teixeira de Moura

Detecção de *DeepFakes* a Partir de Técnicas de Visão Computacional e Aprendizado de Máquina

Banca Examinadora:

- Prof. Dr. Anderson de Rezende Rocha
  Instituto de Computação – UNICAMP
- Prof. Dr. João Paulo Papa
  Departamento de Computação - UNESP
- Prof. Dr. Julio Cesar dos Reis
  Instituto de Computação – UNICAMP

A ata da defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 01 de março de 2021

Agradecimentos

As trajetórias da vida não são percorridas sozinho, são necessários além do desejo e comprometimento individual, apoio e pessoas especiais que compartilharão cada conquista.

Agradeço à CAPES[1] e ao Prêmio Microsoft por financiarem esta pesquisa. Além de todo apoio recebido pelo SAE[2], sem os quais a conclusão deste trabalho se tornaria um caminho árduo.

Agradeço ao Prof. Anderson Rocha por todos os anos de orientação, suporte, compreensão e por proporcionar um ambiente entrosado, além de um laboratório excepcional.

Agradeço aos amigos que estiveram presentes nesta jornada e que tornaram essa experiência única e completa. Em especial ao Welverton que esteve presente durante todo o complexo processo de mudança e que se tornou parte da família. Também não poderia deixar de agradecer ao Rafael por toda ajuda, por todas as vezes que me fez reexplicar os meus problemas e soluções até que estivessem compreensíveis, pela disposição em me ouvir e me ajudar a pensar com maior clareza, enfim, por ser um bom amigo.

Por fim, o apoio mais importante desta trajetória. Agradeço a minha família, que esteve ao meu lado em todos os momentos da minha vida. Em especial ao meu companheiro de vida Jonlenes que está comigo em todos os bons e complexos momentos, me suportando, animando e por vezes acreditando mais em mim que eu mesma e a minha filha Yasmin, que sempre foi compreensiva com todas as longas jornadas que dediquei a esta pesquisa e sempre se demonstrou tão empolgada quanto eu em todas as difíceis mudanças em nossa vida.

[1]O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

[2]Serviço de Apoio ao Estudante

Resumo

Com os avanços tecnológicos e o acesso a informações em tempo real, identificar a veracidade do conteúdo consumido tem motivado o desenvolvimento de pesquisas relacionadas a uma nova classe de problemas, conhecidas como notícias falsas. Estendendo essa evolução a diferentes tipos de representações de conteúdo, encontramos uma das mais preocupantes técnicas que vem sendo utilizada: as *DeepFakes*. Nesta técnica, imagens ou vídeos são artificialmente manipulados a fim de alterar a identidade do interlocutor principal, por meio da troca de face entre duas pessoas. Para isto, são aplicadas técnicas de aprendizado profundo clássicas, além de novas arquiteturas que minimizam o tempo e a quantidade de dados necessários para produzir conteúdos sintéticos/falsos com alto grau de realismo. Analisando essa crescente problemática e os impactos sociais decorrentes em áreas políticas, de entretenimento. Desenvolvemos este projeto, que visa analisar como as técnicas de aprendizado de máquina e aprendizado profundo podem auxiliar na detecção das *DeepFakes*. No decorrer deste trabalho, analisamos algumas das principais bases de dados disponíveis, além de desenvolvermos a nossa própria base, composta de vídeos falsos de conteúdo sensível envolvendo apenas mulheres, principal publico alvo de *DeepFakes* para entretenimento adulto. Avaliamos o comportamento de técnicas tradicionais de aprendizado profundo para a tarefa de classificação entre imagens com e sem manipulações faciais, assim como, a influência de características específicas das imagens e o impacto na qualidade do modelo de detecção gerado. A partir das análises iniciais realizadas, desenvolvemos o nosso método base, em que realizamos o ajuste fino das camadas finais da rede, adaptando-a para a tarefa de classificação binária. Avaliamos a performance deste método, o aplicando a diferentes bases de dados, com as quais obtivemos classificações satisfatórios para quase todas as bases. Em seguida, e com o intuito de aumentar a eficiência da classificação, empregamos três métodos: aumentação de dados, espaços transformados de entrada e função de perda tripla. Para avaliar a robustez dos melhores modelos desenvolvidos, realizamos um teste cego, em que nossos modelos foram submetidos a cenários desconhecidos. Com isso, identificamos que um dos nossos modelos, conseguiu ser generalizar e, decidimos comparar nossos resultados com os obtidos pela literatura. Selecionamos os trabalhos mais utilizados para a detecção de *DeepFakes* e avaliamos o desempenho dos modelos disponibilizados para cada método, diante de um cenário desconhecido e concluímos que o método que propomos foi superior à literatura.

Abstract

With technological advances and access to information in real-time, identifying the veracity of the content consumed has motivated the development of research related to a new class of problems known as fake news. Extending this evolution to different types of content representations, one of the most worrying techniques already in use is called DeepFakes. In this technique, images or videos are artificially manipulated to change the identity of the main interlocutor by exchanging faces between two people. For this fake news, classical deep learning techniques are applied, in addition to the development of new architectures that minimize the time and the amount of data provided for synthetic/fake content. Analyzing this growing problem and the social impacts arising in political and entertainment areas, we developed this project, in which we investigates how machine and deep learning techniques can assist in the task of DeepFakes detection. In the course of our work, we analyzed some of the main databases available for this problem, in addition to developing our own database, composed of fake content videos involving only women, the main target audience of DeepFakes for adult entertainment. We evaluated the behavior of traditional deep learning techniques for classifying between images with and without facial manipulations, as well as the influence of specific characteristics of the images and the impact on the quality of the generated detection model. From these analyzes, we developed our base method, in which we fine-tune the final layers of a network, adapting it for the binary classification task. We evaluated this method's performance, applying it to different databases, and we obtained satisfactory detection rate for almost all databases. Then, to increase the classification's efficiency, we employed three methods: data augmentation, transformed input spaces, and triplet loss function. To assess the robustness of the best models developed, we carried out a blind test, in which our models were applied in unknown scenarios. Thus we identified that one of our models managed to generalize, and we decided to compare our results with those from the literature. We selected the most used works for DeepFakes detection and evaluated their performances in the face of an unknown scenario. We concluded that the method we proposed was superior to the literature.

Lista de Figuras

1.1 Imagens com manipulação facial em filmes. 17
1.2 Evolução das *DeepFake*s com Nicholas Cage. 17
1.3 Comparação entre métodos geradores de *DeepFake*s. 18
1.4 Esquema de funcionamento do bot. 19
1.5 Resumo das publicações relacionadas a *DeepFake*s. 20

2.1 Relação ator-alvo. 24
2.2 Técnicas de Alteração de Face. 25
2.3 *FaceSwap* 26
2.4 *DeepFake* 27
2.5 *DeepFake*s com máscara de segmentação 27
2.6 *DeepFake* com máscara de segmentação e re-encenação 28
2.7 Re-encenação 29
2.8 Representação de Pontos de Referência 30
2.9 Representação de um neurônio. 31
2.10 Representação de uma rede neural. 31
2.11 Rede neural convolucional para classificação de objetos 32

4.1 Cenário geral da metodologia de trabalho proposta. 38
4.2 Método que utilizado para a seleção de vídeos do `DFDC`. 39
4.3 Métodos para Extração de Olhos. 40
4.4 Proposta de Modelo Base. 41
4.5 Proposta para aumentação de dados. 42
4.6 Proposta de classificação com *patches*. 43
4.7 Proposta com abordagens de transformação do espaço de entrada. 44
4.8 Perda Tripla 45
4.9 Tipos de funções de perda triplas 45

5.1 Base de dados `Sensitive`. 49
5.2 Base de dados`Notre Dame Synthetic Face`. 49
5.3 Base de dados `FullTIMIT`. 50
5.4 Base de dados `Face Forensics Plus Plus`. 51
5.5 Base de dados `DFDC`. 52
5.6 Base de dados `DFD - Google`. 52

6.1 Visão Geral dos Experimentos. 54
6.2 Múltiplas faces em um Quadro. 55
6.3 Faces não Detectadas. 56
6.4 Representação do $\overline{SSIM}$ na Face. 57
6.5 Resultado do processo de extração. 58

6.6 Representação UMAP para perda tripla. 60
6.7 Área sob a curva 60
6.8 Arquitetura inicial para o experimento base. 63
6.9 Comparativo entre Modelos com e sem NDSF. 66
6.10 Configuração de camadas para o DFDC. 68
6.11 Representação de Rede para *Patches*. 74
6.12 Mapa de Profundidade. 75
6.13 Esquema para a rede com mapa de profundidade e face. 76
6.14 Comparação entre os métodos para o DFDC. 78
6.15 Variabilidade nos resultados entre as versões base do DFDC. 79
6.16 Comparativo entre a classificação em cada base de dados. 79
6.17 Resultado da predição do DFD - Google. 80
6.18 Matriz de confusão para a predição do DFD - Google - modelo FullTIMIT. 80
6.19 Matriz de confusão para a predição do DFD - Google - modelo Sensitive. 81
6.20 Matriz de confusão para a predição do DFD - Google - modelo FFpp. . . . 82
6.21 Matriz de confusão para a predição do DFD - Google - modelo DFDC. . . . 82
6.22 Matriz de confusão para a predição do DFD - Google - modelo com todas as bases. 83
6.23 Comparação de Resultados com a Literatura. 84

Lista de Tabelas

3.1 Sumário das técnicas da literatura. 36
3.2 Resultado em AUC da classificação pelas técnicas da literatura. 36

5.1 Resumo dos dados das bases de dados 53

6.1 Melhores resultados para o `FullTIMIT`. 63
6.2 Melhores resultados para o `Sensitive`. 64
6.3 Melhores resultados para o `Face Forensics plus plus`. 65
6.4 Melhores configurações para o topo da rede. 68
6.5 Experimento com diferentes arquiteturas. 69
6.6 Otimizador e Taxa de Aprendizado. 70
6.7 Melhores resultados de otimizador e taxa de aprendizado. 71
6.8 Experimentos variando a taxa de congelamento da rede, ou seja, quanto da rede será utilizado para o o ajuste do topo da rede. Em destaque, o melhor resultado obtido. 71
6.9 Experimentos com Aumentação dos Dados. 73
6.10 Experimentos utilizando *Patches* - primeira abordagem. 74
6.11 Experimentos utilizando *Patches* - segunda abordagem. 75
6.12 Resumo dos experimentos realizados com a perda tripla. Em destaque a o resultado da melhor acurácia. 77

A.1 Resultados dos experimentos com o `FullTIMIT`. 102
A.2 Resultados dos experimentos com o `Face Forensics plus plus`. 103
A.3 Experimentos realizados para definir a melhor configuração para o topo da rede. Em destaque, o melhor resultado obtido. 104
A.4 Resumo dos experimentos realizados com a perda tripla. Em destaque a o resultado da melhor acurácia. 105

Lista de Abreviações e Siglas

ACC	*Accuracy*
AUC	*Area Under the ROC Curve*
CGI	*Computer Graphic Imagery*
CNNs	*Convolutional Neural Networks*
DFD	*DeepFake Detection*
DFDC	*DeepFake Detection Chalange*
DNNs	*Deep Neural Networks*
DSP-FWA	*Dual Spatial Pyramid for Exposing Face Warp Artifacts*
FDFtNet	*Fake Detection Fine-tuning Network*
FPR	*False Positive Rate*
FPS	*Frames per Second*
FSGAN	*Face Swapping GAN*
FTT	*Fine-Tune Transformer*
FWA	*Face Warping Artifacts*
GANs	*Generative Adversarial Networks*
GAP	*Global Average Polling*
HOG	*Histogram of Oriented Gradients*
HQ	*High Quality*
LQ	*Low Quality*
LRCN	*Long-term Recurrent Convolutional Neural Networks*
MIT	*Massachusetts Institute of Technology*
MLP	*Multi-layer Perceptron*
MSE	*Mean Squared Error*
NDSF	*Notre Dame Synthetic Face*

PSNR	*Peak Signal-to-Noise Ratio*
ROC	*Receiver Operating Characteristic*
SSIM	*Structural Similarity Index Measure*
SVMs	*Support Vector Machines*
t-SNE	*t-Distributed Stochastic Neighbor Embedding*
TNR	*True Negative Rate*
TPR	*True Positive Rate*
UMAP	*Uniform Manifold Approximation and Projection*

Sumário

1 Introdução **15**
1.1 Objetivos . . . 21
1.2 Questões da Pesquisa . . . 21
1.3 Contribuições . . . 22
1.4 Organização da Dissertação . . . 22

2 Conceitos Relacionados **24**
2.1 Manipulação de Faces em Imagem ou Vídeos . . . 24
2.1.1 FaceSwap . . . 26
2.1.2 *DeepFake* . . . 26
2.1.3 Re-encenação . . . 28
2.2 Detecção Facial . . . 29
2.2.1 Pontos de Referência . . . 30
2.3 Aprendizado Profundo . . . 30
2.3.1 Redes Neurais . . . 30
2.3.2 Redes Neurais Convolucionais . . . 31

3 Trabalhos Relacionados **33**
3.1 Detecção Baseada em Aspectos Físicos . . . 33
3.2 Detecção Baseada em Artefatos . . . 34
3.3 Detecção Orientada por Dados . . . 34
3.4 Visão Geral dos Trabalhos Relacionados . . . 35

4 Metodologia Experimental **37**
4.1 Organização dos Dados . . . 38
4.1.1 DeepFake Detection Challenge . . . 38
4.2 Processamento dos Dados . . . 39
4.2.1 Extração de Faces e *Patches* . . . 40
4.2.2 Conjunto de Treinamento e Validação . . . 40
4.3 Método Base: Modelo Pré-Treinado . . . 41
4.4 Método 1: Aumentação de Dados . . . 42
4.5 Método 2: Representações de Entrada . . . 42
4.5.1 *Patches* . . . 43
4.5.2 Transformação do Espaço de Entrada . . . 43
4.6 Método 3: Funções de Custo . . . 44
4.6.1 Entropia Cruzada . . . 44
4.6.2 Perda Tripla . . . 45

5 Conjuntos de dados **47**
5.1 Sensitive . . . 47
5.1.1 Conjunto de Imagens Reais . . . 47
5.1.2 Conjunto de Imagens Sintéticas/Falsas . . . 48
5.2 Notre Dame Synthetic Face - NDSF . . . 49
5.3 FullTIMIT . . . 50
5.4 Face Forensics Plus Plus, FFpp . . . 50
5.5 DeepFake Detection Chalange . . . 51
5.6 DeepFake Detection . . . 52
5.7 Resumo dos Conjuntos de Dados . . . 53

6 Experimentos e Resultados **54**
6.1 Preparação dos Dados . . . 55
6.1.1 Análise do `DFDC` . . . 55
6.1.2 Extração de Quadros . . . 56
6.1.3 Extração de Faces e *Patches* . . . 57
6.1.4 Divisão entre Treino e Validação . . . 58
6.2 Métricas . . . 59
6.2.1 Métricas de Avaliação . . . 59
6.2.2 Projeção e Aproximação de Variedade Uniforme . . . 62
6.2.3 Índice de Similaridade Estrutural . . . 62
6.3 Experimentos base . . . 62
6.3.1 FullTimit . . . 63
6.3.2 Sensitive . . . 64
6.3.3 Face Forensics Plus Plus . . . 65
6.3.4 Notre Dame Synthetic Face . . . 66
6.3.5 DeepFake Detection Challenge . . . 67
6.4 Método 1: Aumentação dos Dados . . . 72
6.5 Método 2: Representações de Entrada . . . 73
6.5.1 Patches . . . 73
6.5.2 Mapa de Profundidade . . . 75
6.6 Método 3: Perda Tripla . . . 76
6.7 Reanálise da Base de Dados `DFDC` . . . 77
6.8 Comparação entre os Métodos e Análise dos Resultados . . . 77
6.9 Teste Cego e Comparação com a Literatura . . . 79
6.9.1 Teste Cego com o DFD - Google . . . 79
6.9.2 Comparação com a Literatura . . . 82
6.10 Considerações Finais . . . 84

7 Conclusões **86**

A Experimentos Completos **101**
A.1 `FullTimit` . . . 101
A.2 `Face Forensics Plus Plus` . . . 102
A.3 Experimentos com o `DFDC` . . . 103
A.3.1 Camadas do Topo da Rede para o `DFDC` . . . 103
A.3.2 Perda Tripla para o `DFDC` . . . 104

Capítulo 1

Introdução

Na era da tecnologia, qualquer informação está ao alcance de um clique, mas como nos asseguramos da veracidade do conteúdo consumido? Podemos, ainda, partir da premissa de que um vídeo traz uma verdade incontestável?

O momento em que vivemos, nos traz infinitas possibilidades. Hoje, temos acesso a mais informações do que jamais tivemos na história da humanidade. Passamos de espectadores para provedores de notícias. Com isso, adquirimos maiores responsabilidades. Seja vinda de um grande jornal ou de um compartilhamento entre amigos em redes sociais, uma informação errada pode trazer grandes impactos sociais.

Quando tratamos de desinformação e o seu consequente impacto entramos em uma nova área de preocupação mundial intitulada como notícias falsas (*Fake news*). Entende-se por notícias falsas a disseminação de conteúdos que não sejam verdadeiros e que foram propositalmente gerados com o intuito de convencer os leitores acerca da veracidade da informação [87].

A disseminação de notícias falsas traz consequências para diversas áreas, podendo influenciar o mercado financeiro, a vida pública e pessoal dos envolvidos, ocasionar tensões políticas, afetar o desenvolvimento de pesquisas, entre outras [108].

Apesar de não ser recente, notícias falsas produzidas em outras épocas provocavam menor impacto, pois era factível identificar a fonte da notícia e, portanto, confiar em conteúdos gerados por redes de notícia confiável. Essa premissa não é mais verdadeira.

Enquanto a confiança na mídia de massa e nas instituições estabelecidas está diminuindo, o crescente aumento no consumo de conteúdo digital fez com que outros meios de comunicação como blogs, mídia social (como o *Facebook* e o *Twitter*), e-mails, *Podcast* e rádio, se tornassem fontes importantes para a distribuição de informações e, consequentemente, notícias falsas [87]. A facilidade com que as informações são transmitidas acelerou a velocidade e o alcance das notícias falsas.

Além disso, com os avanços tecnológicos, uma gama de novos estilos de desinformação foi criada. Dentre eles temos *Clickbait*, sátiras ou paródias, títulos enganosos, propagandas, notícias tendenciosas, jornalismo com erro, notícias manipuladas, notícias fabricadas e conteúdo patrocinado [34]. Ainda neste seguimento, podemos dividir o tipo de conteúdo multimídia gerado em texto, áudio, imagens e vídeos[11].

Neste trabalho, daremos ênfase nas notícias falsas geradas em imagens ou vídeos. Em particular, focaremos em um novo tipo de tecnologia conhecida como *DeepFake* - técnica

baseada em aprendizado de máquina que proporciona uma ampla variedade de métodos para troca e manipulação de faces, incluindo a utilização de tecnologias de ponta da visão computacional e aprendizado profundo [99].

A primeira aparição de *DeepFake* ocorreu em novembro de 2017, por meio da plataforma de mídia social *Reddit*. Um usuário anônimo conhecido como *u/deepfakes* foi responsável por aplicar uma técnica capaz de trocar a face entre indivíduos, que neste caso, teve o intuito de produzir conteúdo pornográfico [11, 69]. A partir daí, *DeepFake*s passaram a ser utilizadas para gerar vídeos de figuras públicas como celebridades e políticos. Estes foram os principais alvos da técnica devido à quantidade de vídeos e imagens disponíveis publicamente, o que contribuía para a geração de *DeepFake*s.

A disseminação de conteúdos *DeepFake*s é facilitada pela interação entre usuários em fóruns e comunidades, como no *Twitter*, *YouTube*, *Reddit*, *GitHub* e *4chan*. Por isso, surgiram diferentes áreas de aplicação para *DeepFake*, que podem ser divididas em: uso político, entretenimento e cinematografia e entretenimento adulto.

Uso na política. Alguns exemplos de vídeos *DeepFake* disponíveis ilustram o poder potencial dos vídeos como uma ferramenta para conduzir operações políticas. Um exemplo de vídeo falso, produzido usando *FakeApp*, mostra as expressões do rosto do ator Jordan Peele inseridas em um vídeo do Barack Obama (ex-presidente dos Estados Unidos) [11].

Recentemente, vídeos de personalidades políticas tem surgido, dentre eles o do Presidente Richard Nixon anunciando que a missão *Apollo* 11 havia falhado. Este foi criado pelo *Center for Advanced Virtuality do Massachusetts Institute of Technology (MIT)*, em que arquivos do evento real foram utilizados para replicar os movimentos e expressões faciais. Já para o áudio, foi utilizada a voz de um ator para modificar o discurso original [17].

Além deste, várias outras *DeepFake*s de personalidades políticas foram disseminados, como no caso de Nanci Pelosi, porta-voz da Câmara dos Representantes dos EUA, realizando um discurso supostamente bêbada. Hillary Clinton, durante o debate para as eleições presidenciais dos EUA. E Donal Trump, ex-presidente dos EUA, anunciando que irá renunciar ao cargo, sendo um entre várias *DeepFake*s criadas com ele [19, 88, 107].

Entretenimento e Cinematografia. Técnicas para substituição da face de um personagem vem sendo exploradas pelo cinema ao longos dos anos. Entretanto, são comumente utilizados métodos de computação gráfica para a edição dos vídeos. Com isso, além de ser necessário muito tempo para produção, também é preciso uma extensa animação e pós-processamento.

A utilização de *DeepFake*s proporciona novas possibilidades. Com a qual, cenas em que representam risco para o ator podem ser virtualmente adicionadas, além de poder ser realizada a comparação temporal do ator (aparecer mais jovem ou mais velho) sem a necessidade de outra pessoa. Ainda nesse sentido, atores que já faleceram poderiam ser inseridos em novos filmes [80], e também poderia ser utilizado em documentários, para preservar a identidade do indivíduo, mas mantendo as expressões faciais [100].

Na Figura 1.1 apresentamos uma comparação relacionada à evolução da troca de faces para o cinema. Na primeira imagem, a troca foi realizada utilizando um dublê e para

garantir que as diferenças não fossem perceptíveis, as cenas foram gravadas com pouca iluminação (gerado em 1994). Para a imagem seguinte, foi criado um modelo facial por meio de imagens gráficas (*Computer Graphic Imagery*, CGI) obtidas de cenas antigas e, para ser inserido na cena, utilizam-se pontos de referência nos atores substitutos. Por último, temos uma aplicação amadora de *DeepFake* em um trecho de filme.

(a) (b) (c)

Figura 1.1: Imagens com manipulação facial aplicadas a cenas de filmes. (a) Produzida em 1994 para o filme O Corvo, (b) Velozes e Furiosos, 2017 e (c) imagem produzida para o filme Hannibal — criação amadora. Adaptadas de Face [31], Looper [71], SCHAEFER [103].

Também aproveitando trechos de filmes, foram criadas as primeiras *DeepFake*s para entretenimento. Nestas, eram realizadas alterações de cenas de filmes em que os atores tinham suas faces trocadas por faces de outros atores. Nesse sentido, uma coletânea de vídeos foi criada utilizando o ator Nicholas Cage [129]. Selecionamos, na Figura 1.2, imagens de *DeepFake*s comparando a técnica a partir de 2018.

Figura 1.2: Imagens disponíveis na internet mostrando o avanço de *DeepFake*s aplicadas ao ator Nicholas Cage. A primeira linha contêm imagens de 2018, em seguida, 2019 e, por último, 2020. Adaptadas de Bake [5], Derpfakes [20, 21], MitFake [78], Usersub [129], Vanceagher [130].

Aplicativos como o *Zao* [15] e *REFACE* [82] permitem que usuários sem conhecimentos técnicos possam trocar seus rostos com os de estrelas de cinema e se inserir em filmes e clipes de TV. Entretanto, a produção de *DeepFake*s mais convincentes requer conhecimento

técnico, poder de processamento e tempo, como para o *FakeApp*[1], DeepFaceLab [91] e *FaceSwap*[23].

Apesar de *DeepFake*s ainda não serem declaradamente utilizados no cinema, um método de *DeepFake*s para imagens realistas em alta resolução está sendo desenvolvido por pesquisadores dos estúdios Disney em parceria com a Universidade de Zurique [80] (conforme Figura 1.3). Entretanto, ainda produzem imagens falhas para cenas com determinadas expressões ou posições faciais.

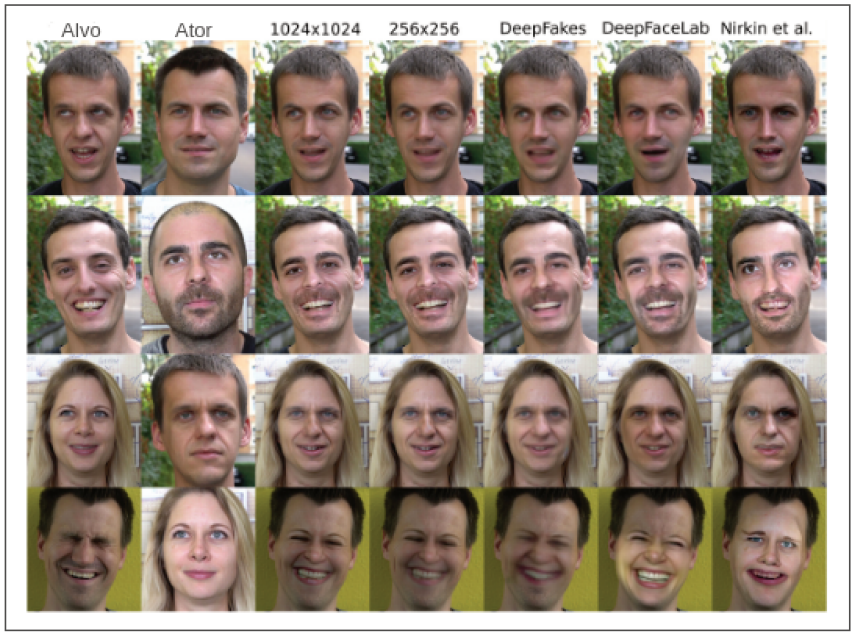


Figura 1.3: Imagem comparativa entre algumas técnicas para geração de *DeepFake*s e a desenvolvida para alta resolução (1024×1024 e 256×256). Adaptado de Naruniec et al. [80].

Uso para entretenimento adulto. Dados apresentados por Ajder et al. [2] relatam que em setembro de 2019, 96% das *DeepFake*s disponíveis na web eram de conteúdo pornográfico. Ainda nesse mesmo trabalho foi identificado que 99% desses vídeos são de personalidades de entretenimento do sexo feminino mostrando um lado muito sombrio e misógino ligados à tecnologia.

O interesse em utilizar *DeepFake*s como gerador de conteúdo pornográfico é visível quando pacotes de software como o *DeepNude* [113], que permite despir virtualmente uma mulher, teve de ser retirado da internet poucos dias após o seu lançamento (estima-se que foram realizadas 545.162 visitas em seus 4 dias de disponibilidade) [2]. Recentemente, um bot para o aplicativo Telegram, baseado em uma versão do *DeepNude*, foi desenvolvido e substitui mulheres vestidas por fotos com nudez [110]. Este processo é descrito na Figura 1.4.

Segundo a pesquisa desenvolvida por Ajder et al. [3], além de serem utilizadas imagens de mulheres sem que haja o consentimento, as imagens geradas pelo bot também estão sendo utilizadas como método de extorsão ou para envergonhar o alvo por meio da

[1]Este software não está mais disponível.

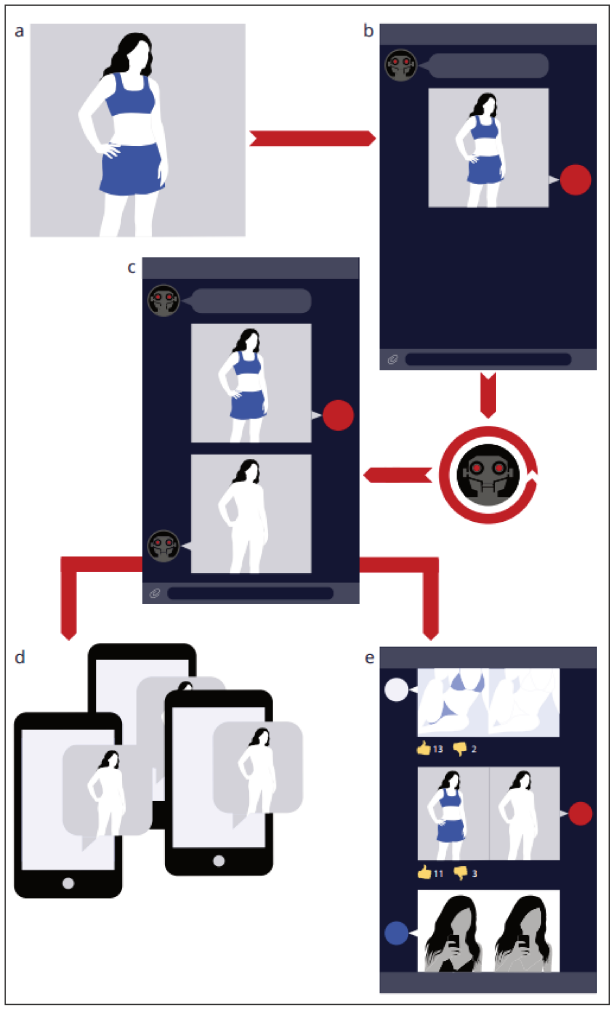


Figura 1.4: Esquema de funcionamento do bot. a) é realizado o *download* de uma imagem da mulher alvo, b) essa imagem é adicionada ao bot do Telegram, c) o bot processa essa imagem e gera a sua versão nua, d) a nova imagem é compartilhada em grupos privados e e) é compartilhada entre canais do Telegram. Adaptado de Solsman [110].

publicação em mídia social ou o envio para pessoas a ela relacionadas.

O crescente interesse em *DeepFake*s fez com que técnicas mais avançadas fossem desenvolvidas, apresentando melhoria na qualidade visual, e no tempo necessário para a produção [96]. Parte disso se deve à utilização de Redes Generativas Adversarias (*Generative Adversarial Networks* - GANs) [37] — método capaz de desenvolver faces sintéticas realistas —, que além de proporcionarem melhorias nas técnicas convencionais, abriram espaço para que GANs específicas para *DeepFake* fosse criadas, como a *Face Swapping GAN* (FSGAN) [85] e a RSGAN [81].

Devido a isso, e considerando o impacto que *DeepFake*s podem gerar, iniciaram-se o desenvolvimento de pesquisas para detectar vídeos com manipulações faciais [34]. Como apresentado na Figura 1.5, devido as *DeepFake*s serem recentes, nos dois primeiros anos após sua aparição pouco se sabia sobre a técnica e como esta poderia evoluir.

Já em 2020, ano em que grande parte das contribuições foram desenvolvidas, o lançamento do *DeepFake Detection Challenge* (DFDC) [47] foi responsável por impulsionar

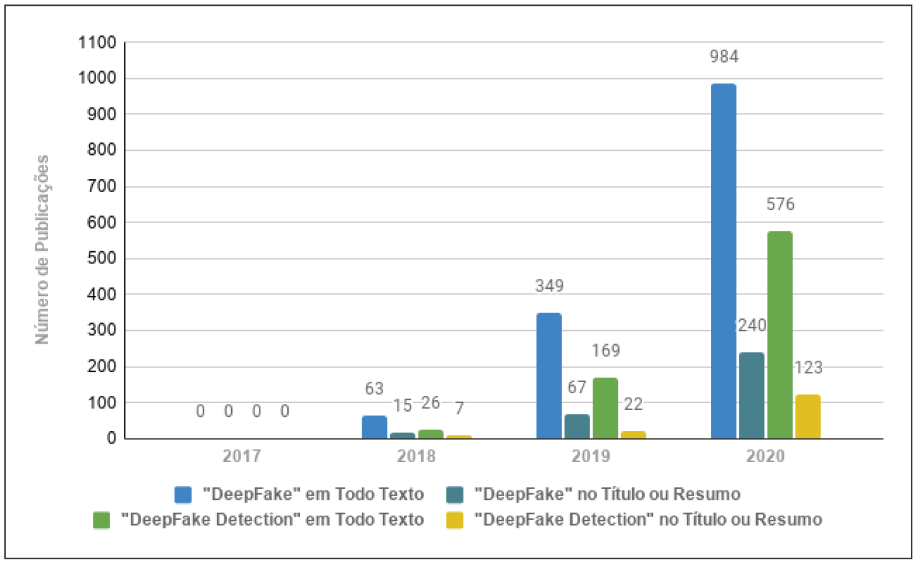


Figura 1.5: Resumo das publicações relacionadas ao tema. Dividimos as buscas considerando a palavra chave escolhida contida em todo o texto do artigo ou apenas no título e resumo. Dados extraídos de Dimensions [25].

diversos grupos ao redor do mundo para que se unissem na tentativa de encontrar uma solução para o problema. Além disso, o desafio também disponibilizou uma base de dados mais representativa, com diferentes técnicas de *DeepFake*s, variação de cenários, posição da câmera, iluminação, entre outras características.

Na busca por uma solução robusta, foram exploradas características baseadas nos aspectos físicos [68, 90, 134], na detecção de artefatos deixados pelo processo de produção das *DeepFake*s [67, 137] e em como os dados são apresentados para a rede [1, 45, 83, 99, 127]. Para os trabalhos que desenvolveram técnicas considerando cenários conhecidos (mesma base de dados para treinamento e teste do modelo), as soluções classificaram satisfatoriamente os dados. Entretanto, quando consideramos a detecção em cenários desconhecidos, os métodos apresentam resultados divergentes. Por isso, procuramos desenvolver um método eficaz em ambos cenários.

No início do desenvolvimento desta pesquisa, coletamos informações sobre as principais técnicas utilizadas para a geração de *DeepFake*s, por meio da seleção de trabalhos com aprimoramentos específicos para esta técnica. Além disso, a partir do monitoramento de sites como *Twitter* e *Reddit*, identificamos e analisamos as ferramentas mais utilizadas pela comunidade. Adicionalmente, realizamos testes com essas ferramentas, visando entender como as *DeepFake*s são criadas e, com isso, melhor direcionamos nossa linha de pesquisa.

Após adquirido este conhecimento, iniciamos as buscas por conjuntos de dados específicos de *DeepFake*s. Nos mantivemos continuamente em busca de melhores conjuntos de dados, com o intuito de acompanhar a evolução da técnica e proporcionar modelos mais robustos. Adicionalmente, avaliamos as tecnologias existentes e suas derivações para detectar *DeepFake*s, bem como sua capacidade de generalização frente a diferentes formas de criação de conteúdo. Por fim, buscamos apresentar um novo método de detecção de *DeepFake*s. Para isto, iremos classificar imagens suspeitosamente falsas, especificamente com adulteração na face, utilizando técnicas de aprendizado profundo. Nossos resultados

consideram além das arquiteturas escolhidas e seus parâmetros, as especificidades de cada conjunto de dados.

Dentre os métodos utilizados, aplicaremos algumas aumentações nos dados, como rotação, brilho e espelhamento, com a qual buscamos entender se introduzir alterações diretas nas imagens originais fornece maior robustez à classificação. Utilizaremos regiões específicas da face: olhos, nariz e boca, que serão combinadas em busca de encontrar a melhor configuração de informações a fim de auxiliar os resultados obtidos considerando a detecção intra base de dados. Ainda com o mesmo intuito, apresentamos a inserção de mapas de profundidade, com o qual buscamos identificar se a construção em profundidade difere quando considerado imagens reais versus falsas. Finalizamos a etapa de experimentos intra base de dados considerando a divisão dos dados no hiper espaço, a partir da utilização da função de custo de perda tripla.

1.1 Objetivos

Nosso principal objetivo é detectar a presença de *DeepFake*s em imagens, por meio de um modelo desenvolvido a partir de técnicas de aprendizado profundo. Como algumas bases de dados são compostas por vídeos, estes deverão ser convertidos em imagens independentes, desconsiderando qualquer característica temporal e relação entre elas. Também realizaremos uma análise detalhada da base `DFDC` a fim de identificar características que suportem a tomada de decisões e que nos auxiliem na exclusão de dados com *DeepFake*s apenas no áudio.

No decorrer do nosso trabalho, cada face será classificada entre real ou falsa, sendo as imagens falsas consideradas como *DeepFake*s. Para a classificação intra base de dados, utilizaremos técnicas para a detecção de falsificações por meio da seleção de arquiteturas de rede que serão utilizadas no treinamento de um modelo. Buscaremos aplicar diferentes técnicas para incrementar os dados utilizados, por meio da aplicação de aumentações e de outras representações de entrada como *patches* e mapa de profundidade.

Para cada base de dados, os modelos que obtiverem os melhores resultados, serão utilizados para a predição de uma base desconhecida, a fim de verificar o desempenho da classificação quando aplicada a técnicas distintas. Além disso, pretendemos verificar a robustez do nosso modelo final, definido após o teste inter base de dados, para comparar com as principais técnicas da literatura e avaliar a evolução da detecção de *DeepFake*s.

1.2 Questões da Pesquisa

Formulamos as principais questões a que este projeto se propõe responder, de modo a nos guiar durante seu desenvolvimento.

I. As bases de dados existentes são suficientes para o desenvolvimento de modelos robustos para detecção de *DeepFake*s?

II. É possível gerar um modelo capaz de detectar *DeepFake*s produzidas com diferentes técnicas de geração de *DeepFake*s? Em outras palavras, é possível atingir generali-

zação na detecção de tais operações de falsificação, sendo invariante ao modo como a falsificação foi produzida?

III. Unir bases de dados construídas com diferentes técnicas para a geração de *DeepFake*s, para produzir um modelo classificador proporciona classificações mais precisas?

1.3 Contribuições

Ao final deste trabalho, apresentamos como contribuições:

I. Base de dados composta por vídeos sintéticos/falsos de celebridades, coletados em site de conteúdo adulto. Esta configura como a única base neste segmento e considera diversas posições da face, possíveis oclusões, diferentes resoluções e variações de iluminação e ambiente.

II. Análise experimental detalhada do conjunto de dados *DeepFake Detection Challenge*, a fim de entender como os dados foram rotulados e desconsiderar *DeepFake*s apenas de áudio.

III. Estudo das principais técnicas para a detecção de *DeepFake*s disponíveis na literatura, além da correlação entre o método de produção de *DeepFake*s e a sua detecção.

IV. Estudo de diferentes técnicas convolucionais para a classificação binária das faces e análise de desempenho dos classificadores considerando modelos por partes e holísticos.

V. Desenvolvimento de um modelo para a detecção de *DeepFake*s robusto, gerado a partir do treinamento em faces da base de dados do `DFDC`.

VI. Análise da classificação inter base de dados, visando a identificação de pontos de melhoria por classe.

1.4 Organização da Dissertação

Para melhor contextualização desta Dissertação, no Capítulo 2, apresentamos conceitos importantes para a compreensão da nossa proposta de trabalho. Nesta, definimos o que é manipulação facial, como uma face pode ser localizada e conceitos relacionados às técnicas base que serão aplicadas neste trabalho.

Revisamos na literatura os métodos já desenvolvidos para a detecção de *DeepFake*s (Capítulo 3) e, no Capítulo 4, descrevemos nossa metodologia de trabalho e como utilizaremos cada base de dados. Além disso, apresentamos as técnicas para extração e detecção facial, como organizamos os dados para serem utilizados nos experimentos e apresentamos quais representações dos dados utilizaremos.

No Capítulo 5, apresentamos os conjuntos de dados que decidimos utilizar e partimos para os experimentos (Capítulo 6). Detalhamos os experimentos realizados, as métricas

utilizadas, os resultados obtidos com cada método proposto e analisamos nossos resultados quando apresentado cenários desconhecidos. Ainda neste escopo, comparamos nossos resultados com a literatura, considerando a abordagem com cenários desconhecidos. Finalizamos com o Capítulo 7, em que apresentamos uma sucinta análise dos resultados obtidos e indicamos algumas abordagens para a continuidade deste trabalho.

Capítulo 2

Conceitos Relacionados

Por *DeepFake*s ser um conceito recente, parte importante para o entendimento deste trabalho inclui definir quais os tipos de manipulações de faces existentes. Para isso, apresentamos na Seção 2.1, as definições de alterações de face em arquivos de mídia digital e as principais técnicas utilizadas para a sua geração. Em seguida, como nossa abordagem irá utilizar apenas a face para a nossa avaliação, apresentamos os conceitos de detecção facial na Seção 2.2. Já a Seção 2.3 retrata os conceitos de aprendizado profundo, técnica em que nos baseamos para desenvolver a nossa solução para a detecção de *DeepFake*s.

2.1 Manipulação de Faces em Imagem ou Vídeos

A manipulação de imagens é bastante utilizada na área do entretenimento para melhorar características de uma imagem ou vídeo. Pacotes de *software* para edição de imagens como *Photoshop* já estão em uso há décadas. Recentemente, novas ferramentas de manipulação automática como o *FaceApp* [32] começaram a ser aprimoradas e amplamente utilizadas [135].

A manipulação da face envolve a modificação de atributos faciais, troca ou transformação de duas faces, geração de rostos sintéticos e re-encenação de expressões faciais em imagens ou vídeos [4]. Para melhor entendimento, utilizaremos a nomenclatura ator-alvo, em que ator se refere à face que será inserida no alvo (imagem de destino), conforme exemplificado na Figura 2.1.

Figura 2.1: Relação entre ator e alvo. À esquerda, temos o alvo, no meio, o ator e, por último, o resultado do ator aplicado no alvo. Adaptada de Bitouk et al. [8].

As manipulações faciais em imagens ou vídeos, inicialmente utilizada pela área de processamento de imagens, tem se tornado cada vez mais comum e acessível a qualquer usuário da internet. E, com a utilização de Redes Neurais Profundas *(Deep Neural Networks, DNNs)*, o processo de geração de vídeos falsos convincentes está mais fácil e rápido [69].

Por meio da Figura 2.2, apresentaremos os três principais métodos de manipulação facial (*DeepFake*, *FaceSwap* e Re-encenação), que serão descritos nas subseções a seguir.

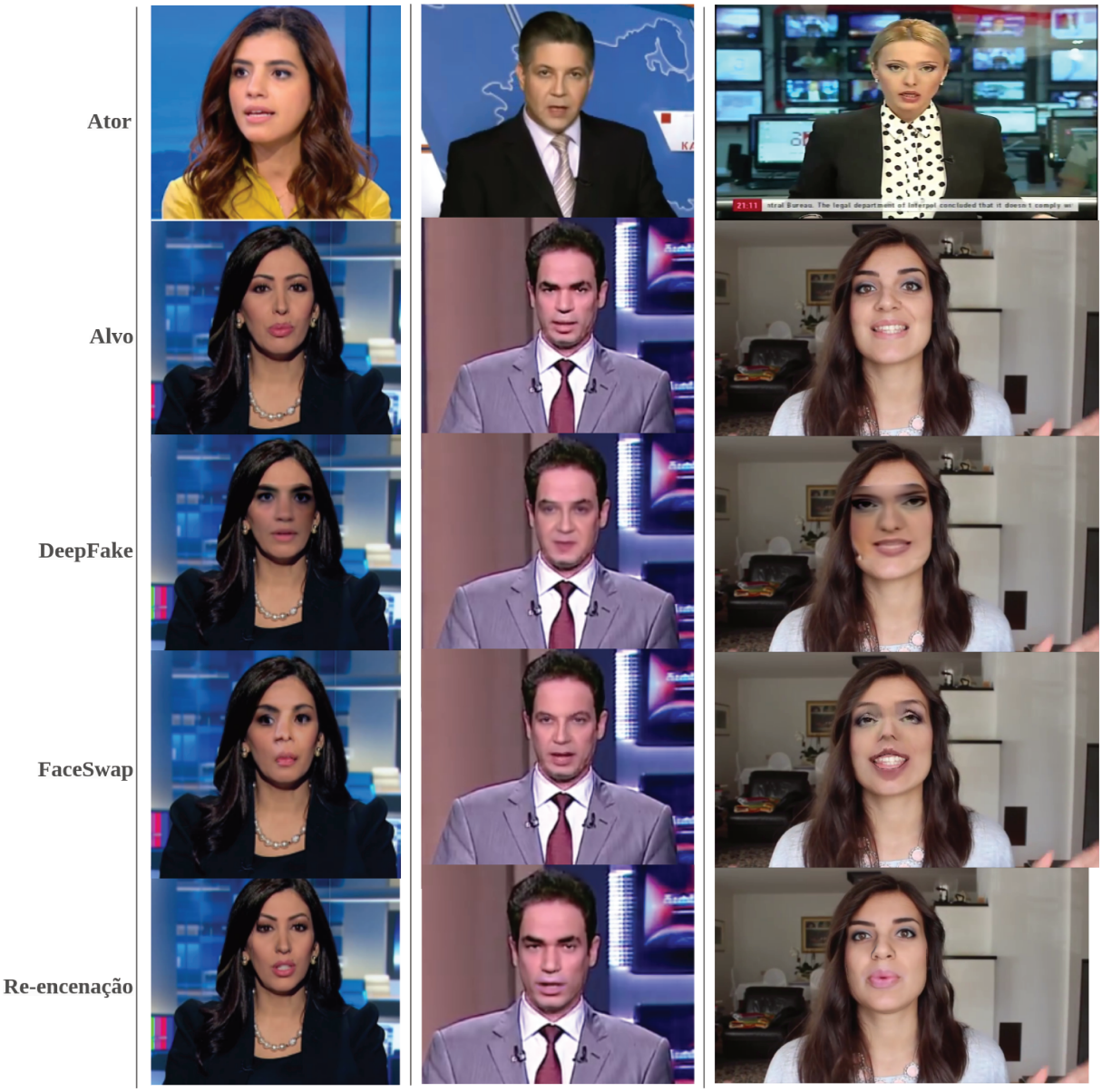


Figura 2.2: Exemplos de alterações de face aplicando a técnica de *Face2Face*, *DeepFake* e *FaceSwap*. A primeira linha refere-se ao ator, em seguida, o alvo e as demais linhas trazem a técnica descrita aplicando ator ao alvo. Imagens obtidas na base de dados `Face Forensics plus plus`.

2.1.1 FaceSwap

FaceSwap é uma categoria de troca de rosto. Nesta, se aplica uma abordagem baseada em técnicas de computação gráfica, que irá transferir a região do rosto de um vídeo de origem para um vídeo de destino. Além disso, pode ser executada em tempo real [99].

A região é extraída com base nos marcos faciais detectados (ver referência na Subseção 2.2.1) e, usando esses pontos de referência, o método se ajusta a um modelo 3D, que é inserido na imagem de destino, minimizando a diferença entre o modelo e os pontos de referência da face de destino. Finalmente, o modelo é combinado com a imagem alvo sendo aplicadas correção de cores e, por vezes, iluminação [90, 99]. Na Figura 2.3, apresentamos parte do processo para geração da imagem falsa ou sintetizada.

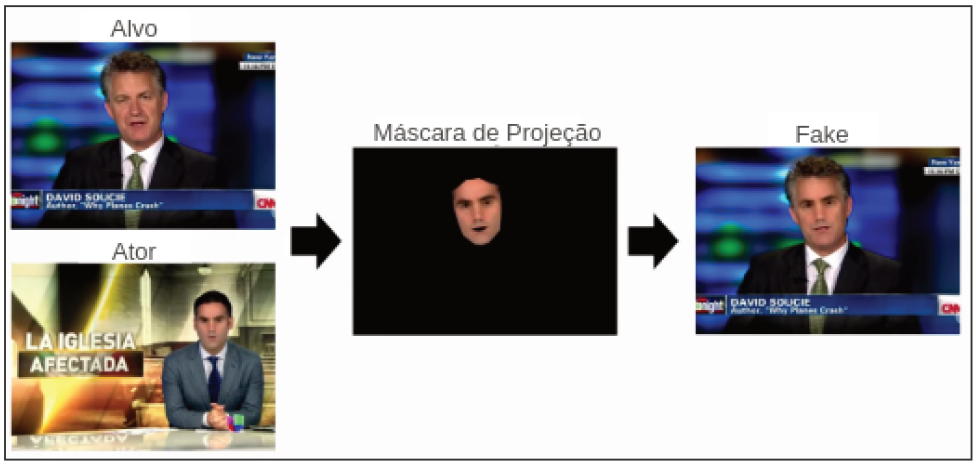


Figura 2.3: Esquema de uma abordagem para gerar *FaceSwap*. A face é detectada em ambas imagens, uma máscara de projeção é gerada a partir da face do ator e é inserida na imagem alvo. Adaptada de Rössler et al. [99].

2.1.2 *DeepFake*

DeepFake é um termo que está relacionado à substituição de faces em imagens ou vídeos originais com base em modelos de aprendizado profundo [134]. Surgiu em 2017, com o primeiro uso registrado sendo para a troca da face de uma celebridade em uma atriz pornô. Desde então, ficou conhecido por ser um método de manipulação compartilhado em fóruns online.

Inicialmente, este método era baseado em dois auto-codificadores (*auto encoders*), treinados para reconstruir imagens de treinamento do ator e do alvo. Isso ocorre por meio de a um detector de rosto utilizado para extrair e alinhar as faces. Para criar uma imagem sintética/falsa, o codificador e o decodificador treinados na face do ator são aplicados à face do alvo. A saída do codificador é combinada com o restante da imagem usando métodos de edição de imagem [99]. Posteriormente, essa técnica foi aprimorada para criar aplicativos como o *FakeApp*, que facilitou o uso de usuários sem conhecimento técnico ou poder computacional [1].

Um dos primeiros métodos de *DeepFake* desenvolvido considerava os pontos de referência da face para extrair a região de interesse do ator e inserir nos mesmos pontos da

face alvo. Para isso, era necessário realizar o alinhamento antes de extrair os marcos e, por meio de uma rede neural, inserir a face no alvo. Em seguida, era feito o realinhamento e ajustes da face para o melhor encaixe, conforme Figura 2.4. Este método foi a base para o desenvolvimento de diversas ferramentas mais modernas [8, 49, 63, 84, 136].

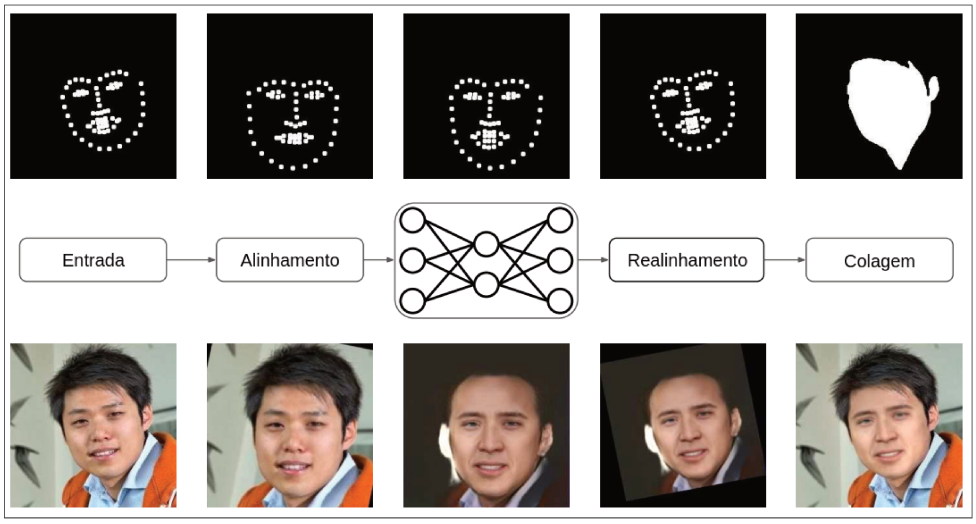


Figura 2.4: Esquema para geração de *DeepFake*s. Adaptada de Korshunova et al. [63].

Com a crescente utilização de *DeepFake*s, a técnica foi sendo aprimorada para que se tornasse invariante a oclusões da face por óculos, barba ou parte do cabelo. Em trabalhos como o de Nirkin et al. [84], uma etapa de segmentação da face é aplicada após o alinhamento e resulta em uma máscara com o formato da face tanto do ator como do alvo. Então, a face do ator é aplicada no alvo, com o auxílio da máscara extraída (Figura 2.5).

Figura 2.5: Esquema de geração para *DeepFake*s aplicando máscara de segmentação. Adaptada de Nirkin et al. [84].

Abordagens de *DeepFake*s utilizando GANs se tornaram o estado da arte, com redes desenvolvidas especificamente para troca de faces. A FSGAN [85] pode ser utilizada para criar *DeepFake*s sem que haja a necessidade de treinamento nas faces escolhidas. Isso ocorre devido a: um gerador de re-encenação (G_r) e um de segmentação (G_s), um gerador de pintura facial (G_c) e um gerador para mesclagem (G_b). Os geradores G_r e G_s geram um mapa de calor [36] que codifica os pontos de referência da face e gera uma imagem

re-encenada mantendo a mesma pose e expressão do alvo. Além disso, é calculado o mapa de segmentação da face e da oclusão (como o cabelo, por exemplo). Para preencher as partes que estavam oclusas, o G_c utiliza a segmentação para estimar os pixels da região. Por fim, o G_b combina o rosto obtido com a imagem alvo. O esquema desta arquitetura pode ser visualizado na Figura 2.6.

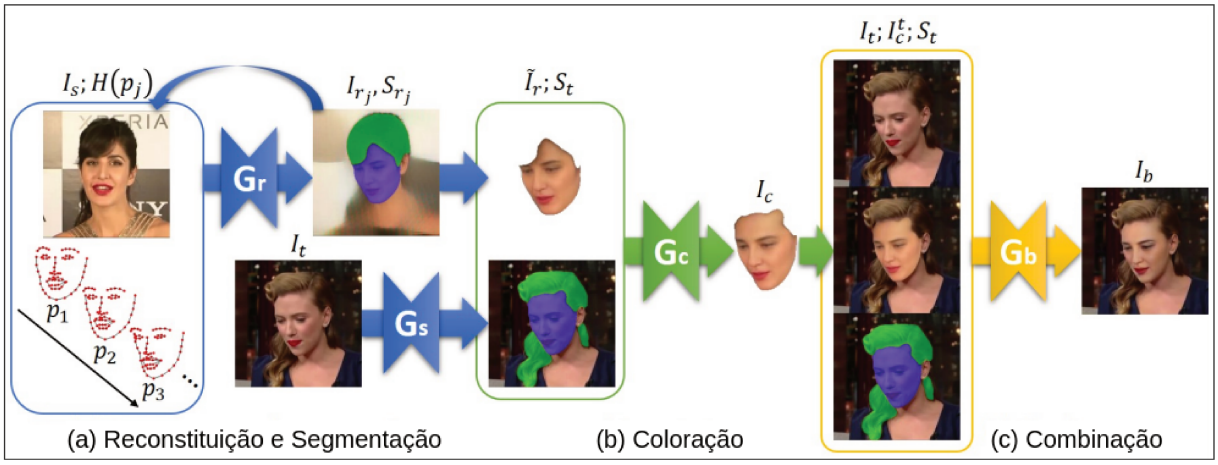


Figura 2.6: A face re-encenada é obtida pelos geradores de reconstituição e segmentação (a), outro gerador complementa a face inserindo as partes omitidas pela oclusão (b) e um último gerador combina a face do alvo e do ator por meio da máscara de segmentação (c). Adaptada de Nirkin et al. [85].

2.1.3 Re-encenação

A re-encenação engloba dois tipos de manipulação. A primeira é uma técnica de manipulação da expressão facial comumente chamado de *Face2Face*, que permite a transferência de expressões faciais de um ator para um alvo em tempo real. Além disso, temos auto-representação, em que aplicamos a face obtida com o *Face2Face* para representar novamente as expressões faciais de um vídeo de origem [98].

No trabalho de Dang et al. [16] é utilizado *Face2Face* combinado com a transferência da posição 3D da cabeça, rotação, expressão e piscar de olhos de um ator para um vídeo alvo. Além disso, o rosto é animado adicionando um áudio de entrada. Para isso, são necessários alguns minutos de vídeos pré-gravados da pessoa alvo que será treinado para reconstruir seu modelo facial. O programa rastreia as expressões do vídeo do ator para o alvo. A síntese da imagem final é renderizada sobrepondo a face de destino com uma forma de mistura facial transformada para se ajustar à expressão facial de origem [1]. Na Figura 2.7, apresentamos um exemplo de como a re-encenação funciona.

Figura 2.7: Imagem contendo re-encenação. Adaptada de Rössler et al. [99].

2.2 Detecção Facial

A detecção facial consiste no processo de localização de uma face em uma imagem. Essa é uma etapa importante, pois as técnicas abordadas nesse trabalho são focadas na face do indivíduo.

A localização consiste em encontrar uma região retangular que contenha a face. Para isso, diversos métodos podem ser utilizados, dentre eles, *Haar Cascade* [112], Histogramas de Gradientes *(Histogram of Oriented Gradients, HOG)* associado a Máquina de Vetores de Suporte *(Support Vector Machines, SVMs)* [106] e o estado da arte, aprendizado profundo.

O detector de faces baseado em *Haar Cascade* foi o estado da arte por muitos anos desde 2001, quando foi apresentado por Viola e Jones [132]. Já o método baseado em recursos HoG e SVM ainda é utilizado atualmente. Entretanto, esses métodos estão sendo substituídos pelos métodos baseados em aprendizado profundo, principalmente, para obter mais robustez nas detecções (rostos em diferentes ângulos e com oclusões, por exemplo).

O estado da arte, que se baseia em aprendizado profundo, utiliza um método de detecção de objeto de margem máxima (mais detalhes em King [59]). O processo de treinamento para este método é simples, não sendo necessário uma grande quantidade de dados para treinar um detector de objeto personalizado. Este método está implementado e é distribuído pela *Dlib*[1] [60] com um modelo pré-treinado para detecção facial.

Com a região da face localizada, o próximo passo consiste na localização de alguns pontos de referência nessa face, conforme apresentado na próxima seção.

[1]Biblioteca amplamente utilizadas para tarefas de diversas áreas, como aprendizado de máquina, processamento de imagens, entre outros.

2.2.1 Pontos de Referência

Após detectar um rosto em uma imagem, é realizada uma estimativa do ponto de referência do rosto, ou seja, são identificados os pontos-chave (também conhecido como pontos fiduciais) de um rosto, como olhos, nariz, boca, mandíbula e sobrancelha [9]. Esses pontos se tornam essenciais quando se deseja localizar cada uma das regiões do rosto, para utilizar como regiões de interesse, alinhamento facial, estimativa de pose da cabeça, troca de rosto, entre outros.

Existem diferentes modelos para estimativa de pontos de referência da face, dentre eles, o estado da arte que utiliza um conjunto de árvores de regressão (mais detalhes em Kazemi e Sullivan [48]). Independente da técnica, a saída consiste em um conjunto de pontos, conforme o exemplo apresentado na Figura 2.8, onde temos um conjunto de pontos para cada uma das regiões da face.

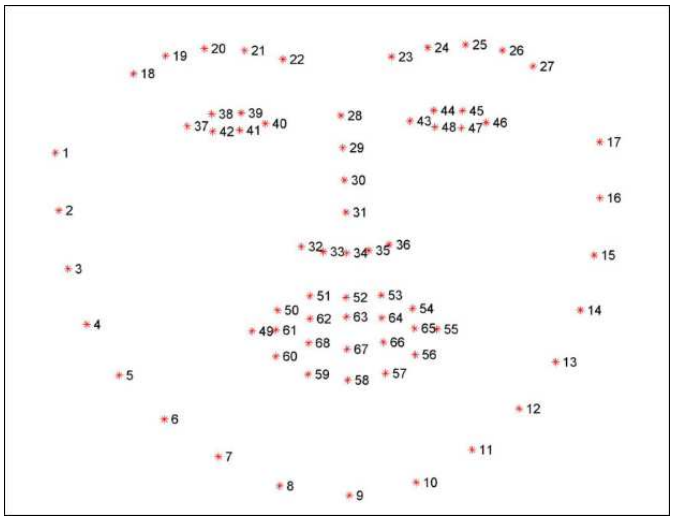


Figura 2.8: Representação numérica de 68 pontos de referência da face. Adaptada de Rosebrock [97].

2.3 Aprendizado Profundo

Aprendizado profundo é uma subárea de aprendizado de máquina que utiliza estratégias para gerar hierarquias profundas não lineares que irão extrair características dos dados e transformá-las em diferentes representações [22]. Para isso, utiliza como base (no caso de imagens) as Redes Convolucionais, apresentada na Seção 2.3.2, derivada das Redes Neurais, como apresentado na Seção 2.3.1.

2.3.1 Redes Neurais

As redes neurais são inspiradas no comportamento do cérebro humano, ou seja, como a sinapse — sinal que é transferido de um neurônio a outro — é realizada. A comunicação entre os nós (neurônios) é iniciada quando há entradas suficientes. Essa ativação se propaga através da rede, criando em resposta o resultado.

Na prática, em cada nó, os valores de entrada são multiplicados pelos seus respectivos pesos (utilizados para que a rede possa aprender), somados e aplicados a uma função de ativação, que define se o neurônio será ativo ou não. Um exemplo desse processo é apresentado na Figura 2.9.

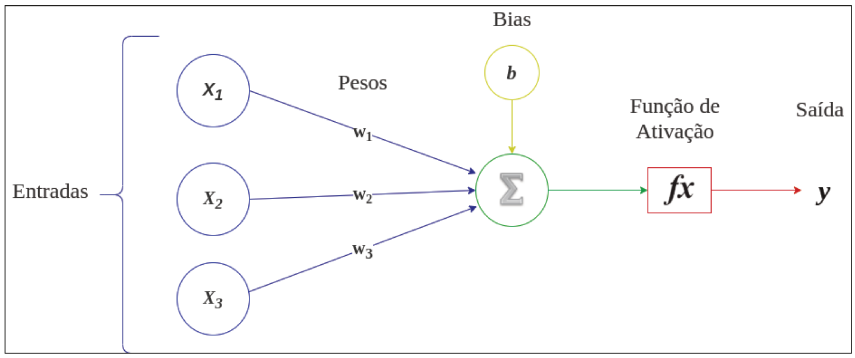


Figura 2.9: Representação de um neurônio, em que x_i representa as entradas, w_i são os pesos, b é o bias, f_x é a função de ativação e y é a saída.

Uma rede neural pode então ser definida como uma coleção de nós conectados entre si. Conforme exemplo apresentado na Figura 2.10, a primeira camada é a entrada, as intermediárias são camadas ocultas e a última é a saída da rede. As redes neurais usam uma hierarquia em camadas. Para isso cada camada aprende com a anterior e repassa sua saída para a próxima, finalizando ao fornecer um valor de saída que pode ser contínuo ou categórico.

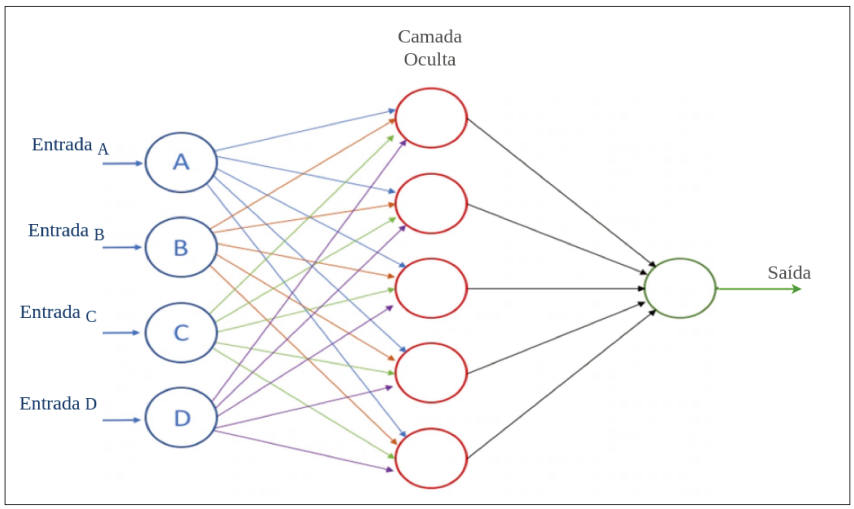


Figura 2.10: Representação de uma rede neural. Adaptada de Tierney [126]

Essa redes são chamadas profundas quando aumentamos a quantidade de camadas ocultas nas quais os dados são transformados e pela capacidade de aprender diferentes características.

2.3.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais *(Convolutional Neural Networks, CNNs)* são compostas por tipos de camadas, como: entradas, convolução, agrupamento, totalmente conectadas,

saída, entre outras e, operações de exclusão *Dropout*, normalização em lote (*Batch Normalization*) e deconvolução. Cada uma com um propósito específico, como sumarização ou resumo, conexão ou ativação.

A ideia dessas redes é aprender representações diretamente das imagens, onde as camadas (ou filtros) convolucionais, extraem características globais de baixo nível da imagem e as camadas completamente conectadas (baseadas nas Redes Neurais) aprendem características específicas da aplicação para qual a rede está sendo treinada — baseadas nas características de baixo nível da imagem, conforme o exemplo da Figura 2.11. A propagação de erro e aprendizado dos parâmetros ocorre por meio de algoritmos baseados no Retro-propagação (*Back-Propagation*) [66].

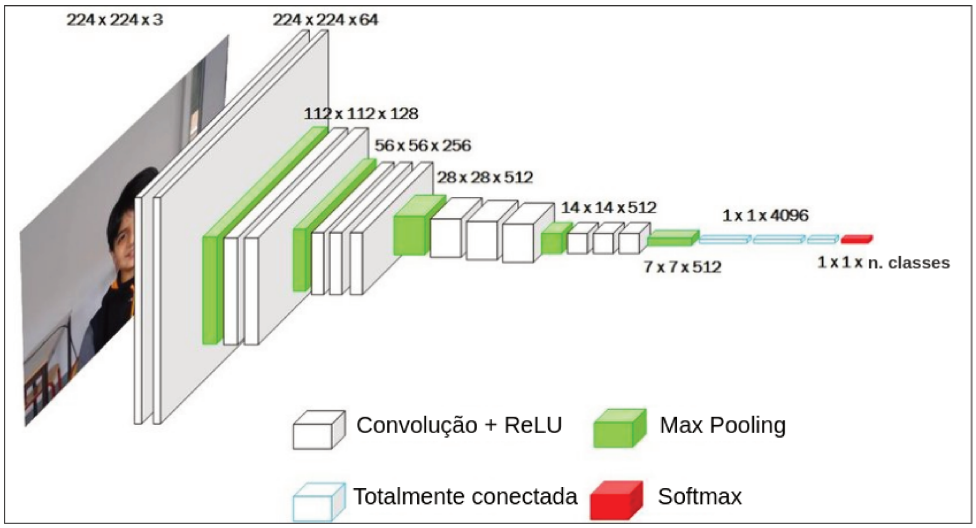


Figura 2.11: Rede neural convolucional para classificação. Adaptada de [58].

Uma das principais características das CNNs é a capacidade de generalização — especialmente nas camadas iniciais — fazendo com que essas redes possam ser adaptadas para outros conjuntos de dados ou aplicações. Para muitos problemas, não é possível treinar uma CNN completa, já que essas redes convolucionais exigem muitos dados rotulados para a otimização de seus parâmetros. Por isso, diversos trabalhos fazem uso da capacidade de generalização e extraem características por meio de CNNs pré-treinadas com outros conjuntos de dados e as adaptam para o problema específico [10].

Quando um modelo de aprendizado de máquina é treinado, o que realmente ocorre é o ajuste dos parâmetros de modo que possa ser mapeada uma entrada específica (como por exemplo uma imagem) para uma saída (rótulo). O objetivo da otimização é encontrar o ponto mínimo de uma função de custo/perda. Essa função mede o quão bem a rede está mapeando os dados de entrada na saída em comparação com os rótulos previamente conhecidos para cada imagem [114]. Para o nosso trabalho, exploraremos diferentes CNNs para o problema de detecção de *DeepFake*s.

Capítulo 3

Trabalhos Relacionados

Nos últimos anos, o processo de geração de *DeepFake*s foi continuamente aprimorado por diversos pesquisadores [26, 29, 30, 62, 70, 98]. Desde o trabalho de Pavel Korshunov [90], as falsificações melhoraram a qualidade visual e os métodos utilizados em sua geração se tornaram mais complexos. Atualmente, as técnicas mais utilizadas são baseadas em GANs [81, 85], com manipulações apresentando alto grau de realismo [80]. Ao passo que novos e melhores *DeepFake*s foram produzidas, surgiu a preocupação com os efeitos sociais de tal técnica [2, 3, 11, 88, 100]. Claypoole [14] descreve a preocupação com o impacto provocado por conteúdos audiovisuais manipulados perante os tribunais judiciários, uma vez que evidências digitais são aceitas como prova, muitas vezes determinante. A partir disto, iniciaram-se pesquisas buscando encontrar métodos eficazes para identificar cenários em que possíveis falsificações faciais foram inseridas.

A fim de direcionar nossa pesquisa e definir a melhor abordagem para a detecção de *DeepFake*s, buscamos na literatura quais métodos já estavam sendo empregados. Estes podem ser dividido em três principais abordagens: busca por aspectos físicos, identificação de artefatos e orientada por dados [72], conforme apresentado nas seções a seguir.

3.1 Detecção Baseada em Aspectos Físicos

Imagens sintéticas geradas a partir de GANs, usualmente apresentam inconsistências físicas na representação de aspectos do rosto. Isto possibilita que em trabalhos como o desenvolvido por Li et al. [68], seja explorada a variação que ocorre entre o piscar de olhos de vídeos reais e falsos/sintéticos. Para tanto, é utilizada uma Rede Neural Convolucional Recorrente de Longo Prazo (*Long-term Recurrent Convolutional Neural Networks*, LRCN), que considera aspectos temporais relacionando os estados dos olhos (ao longo do tempo). Isso possibilita identificar inconsistências entre o abrir e fechar dos olhos. Está técnica foi aplicada considerando uma limitação na geração de *DeepFake*s, em que falsificações geradas a partir de imagens, não apresentam uma transição natural para o movimento dos olhos.

De forma semelhante, na técnica empregada por Pavel Korshunov [90] são detectadas inconsistências entre os movimentos dos lábios e a fala no áudio. Para isso, eles se baseiam na sincronização labial, visto que em vídeos autênticos o movimento labial e a fala tendem

a ser sincronizados, o que pode não ocorrer em vídeos adulterados.

Outro tipo de incompatibilidade é abordado por Yang et al. [134]. Nesse trabalho, são comparados os pontos de referência extraídos da imagem real e os da sintética/falsa relacionada. Com esta análise, buscam-se erros entre as poses da cabeça. Sendo a diferença encontrada utilizada como um vetor de características no treinamento de um *SVM* para diferenciar imagens contendo *DeepFake*s de reais.

3.2 Detecção Baseada em Artefatos

Uma característica de algoritmos para a manipulação é a inclusão de artefatos visíveis (desfoque, variação na coloração) ou invisíveis (ruídos) durante o processo de síntese da nova face. Em busca de detectar tais informações, Li e Lyu [67] considera artefatos de distorção da face (*Face Warping Artifacts*, FWA), devido ao fato das imagens serem sintetizadas com tamanho fixo, se torna necessário aplicar uma transformação a fim de ajustá-la ao tamanho da imagem alvo. Uma melhoria do método proposto pelos autores, consiste na pirâmide espacial dupla para exposição de artefatos de distorção facial (*Dual Spatial Pyramid for Exposing Face Warp Artifacts*, DSP-FWA), que simula diferentes resoluções para as imagens, por meio da pirâmide espacial dupla [39]. Essa variação proporcionou uma melhoria significativa na classificação de conteúdos sintéticos.

Analisando outro tipo de artefato, Zhou et al. [137] desenvolveu uma rede de dois fluxos, em que um fluxo detecta inconsistências de baixo nível entre *patches* da imagem e o outro detecta adulterações nos rostos. Especificamente, o primeiro fluxo procura artefatos inseridos durante o processo de geração das *DeepFake*s. Já o segundo emprega esteganálise em regiões da imagem, que realiza a divisão dos exemplos no hiper-espaço por meio de uma função de perda tripla [104].

3.3 Detecção Orientada por Dados

Nesta abordagem, são empregadas diferentes redes de aprendizado de máquina treinadas com dados específicos, de modo a otimizar a solução. Com base nisto, o trabalho desenvolvido por Rössler et al. [99] aplica CNNs, para a detecção de *DeepFake*s. Além disso, este é o único trabalho que realiza um comparativo entre o resultado obtido entre máquina e humanos. Para esta análise, foram extraídas faces considerando a região expandida, ou seja, com alguns *pixels* ao redor da face. Os autores ainda desenvolveram o conjunto de dados `Face Forensics plus plus` que apresenta manipulações com base nos métodos clássicos de computação gráfica: Re-encenação e *FaceSwap*, bem como *DeepFake*s e Texturas Neurais.

Afchar et al. [1] analisam como detectar *DeepFake*s em um nível mesoscópico [95]. Para isso, propuseram duas CNNs: *Meso-4* e *MesoInception-4*, que contém um pequeno número de camadas para a extração de características e uma rede densa para a classificação. O principal foco desta abordagem é a busca por informações inseridas em um nível intermediário, ou seja, que não são visíveis a olho nu mas também não precisam de uma análise microscópica.

Jeon et al. [45] propuseram uma Rede de Ajuste Fino de Detecção Falsa (*Fake Detection Fine-tuning Network*, FDFtNet) que combina o transformador de ajuste fino (*Fine-Tune Transformer*, FTT) que extraí características utilizando módulos de auto-atenção, com uma rede neural convolucional. Com essa abordagem é possível reutilizar modelos pré-treinados em poucas imagens, de modo que permita ao ajuste fino detectar imagens sintetizadas.

Diferentemente dos detectores com CNNs tradicionais, o trabalho de Nguyen et al. [83] utiliza uma rede em cápsula para detectar manipulações tanto em vídeos (por meio da separação em quadros), imagens (divididas em *patches*) ou apenas o rosto. A imagem é pré-processada e passa por uma parte da rede *VGG-19* pré-treinada na base *ILSVRC*. Em seguida, o resultado obtido é utilizado como entrada para a rede em cápsula, que realiza o pós-processamento e a classificação multi classe.

Já a abordagem aplicada por Tolosana et al. [127] considera como representações de entrada: imagem da face segmentada e regiões faciais específicas (*patches*) segmentadas. Então, utiliza as redes *Xception* e Cápsula, para realizar a classificação.

3.4 Visão Geral dos Trabalhos Relacionados

Buscamos identificar as principais características de cada trabalho descrito ao decorrer deste capítulo e reunir as informações referentes às arquiteturas de rede utilizadas. Não pudemos comparar os resultados obtidos na literatura, devido a não uniformidade da métrica empregada. Adicionalmente, identificamos se os autores consideram um cenário intra ou inter base de dados para a classificação. Reunimos essas informações e as apresentamos na Tabela 3.1.

Analisando os trabalhos já realizados, decidimos concentrar nossa pesquisa na detecção de imagens sintéticas/falsas tanto utilizando a abordagem de *FaceSwap* quanto *DeepFake*. Além disso, identificamos que no trabalho de Li et al. [70], diferentes métodos foram utilizados para comparar o comportamento de cada técnica frente às especificidades de cada base de dados utilizada. Dentre os métodos utilizados por Li et al. [70] escolhemos os que obtiveram melhores resultados, para as bases que também utilizaremos em nosso trabalho, conforme descrito na Tabela 3.2. Estes serão utilizados para comparar o resultado que obtivemos na predição da base `DFD - Google` com o obtidos pelos autores para a classificação das imagens com essa mesma base.

Considerando os trabalhos descritos nessa seção, buscamos identificar suas deficiências e explorar características intrínsecas de cada base de dados. Iniciamos aplicando a abordagem considerando características de aprendizado profundo, por meio da escolha de arquiteturas de rede consagradas para a tarefa de classificação.

Dividimos nosso trabalho de modo a selecionar a melhor abordagem para cenários conhecidos (intra) e avaliar a eficácia dos modelos em cenários desconhecidos a partir do teste cego (inter). Além disso, diferentemente dos trabalhos aqui abordados, empregamos *patches* combinados entre si e mapa de profundidade, como representações de entrada completa. Finalizamos alterando a função de custo em que utilizamos uma perda tripla semi-difícil para a divisão dos dados, antes de realizar a classificação final.

Método	Arquitetura	Treinamento	Predição
Piscar de Olhos [68]	VGG16 [54] LRCN [27]	CEW [111] UADFV	UADFV
Movimento Labial [90]	VGG FaceNet [104] PCA LDA IQM [35] SVM	FullTIMIT	FullTIMIT
Posição da Cabeça [134]	SVM [106]	UADFV	DARPA GAN*
FWA & DSP-FWA [67]	VGG16 ResNet50 [57] ResNet101 [57] ResNet152 [57]	Próprio	UADFV [68] FullTIMIT [90, 102]
Dois Fluxos [137]	Inception V3 [116]	SwapMe [137] FaceSwap [137]	FF++
CAP [99]	Meso-4 XceptionNet [55]	FF++ [99]	FF++
Características Mesoscópicas [1]	Meso-4 [1] MesoInception-4 [1]	Próprio	Próprio
FDFtNet [45]	XceptionNet SqueezeNet [44] ShallowNetV3 [117, 118] ResNetV2 [57]	Próprio	Próprio
Cápsula [83]	CapsuleNet	FF++	FF++
CAP [70]	Inception V3 MesoNet [1] SVM ResNet50 XceptionNet CapsuleNet	SwapMe FaceSwap UADFV FF++	UADFV FullTIMIT FF++ DFD [30] DFDC Celeb-DF
CAP [127]	XceptionNet CapsuleNet [101]	UADFV FF++ Celeb-DF [70] DFDC [26]	UADFV FF++ Celeb-DF DFDC

Tabela 3.1: Sumário das técnicas de detecção de *DeepFake*s empregadas pela literatura. CAP: Características de Aprendizado Profundo. Utilizamos a linha tracejada para separar a abordagem aplicada. *Base de dados não especificada pelo autor.

Método	FullTIMIT*	FFpp	DFDC	DFD - Google
FWA	99.9	80.1	72.7	74.3
DSP-FWA	99.9	93.0	75.5	85.9
Xception-raw	56.7	99.7	49.9	53.9
Cápsula	78.4	96.6	53.3	64.0

Tabela 3.2: Resultado em AUC da classificação de algumas bases de dados pelas principais técnicas de detecção de *DeepFake*s disponibilizadas pela literatura. *Para a base `FullTIMIT`, utilizamos os resultados obtidos para as imagens em baixa resolução. Adaptado de [70].

Capítulo 4

Metodologia Experimental

Impulsionadas pelo crescente volume de falsificações e bases de dados *DeepFake*s, as técnicas de detecção evoluíram bastante nos últimos anos. Se aproveitando das diversas características do processo de geração e/ou da aparência das faces, obtiveram resultados interessantes no cenário intra, ou seja, treinados e avaliados em conjuntos de imagens com características semelhantes ou gerados pelas mesmas técnicas. Porém, em cenários inter, o desempenho destas técnicas ficam aquém do esperado.

Com isso em mente, nosso trabalho consiste em analisar as bases existentes, assim como as técnicas convolucionais e arquiteturas que melhor se adéquam à classificação das imagens em reais e sintéticas/falsas. Ainda neste intuito, avaliamos os efeitos de aplicar aumentação nos dados, normalização e diferentes funções de custo. Avaliamos como o modelo se comporta com espaços de entrada distintos e ao uso de transferência de aprendizado. Além disso, exploramos a utilização de modelos por partes e holísticos. Finalizando com a comparação de comportamento do modelo diante de cenários intra e entre base de dados.

Representamos a ideia central da nossa solução na Figura 4.1. Utilizaremos o conjunto de faces (dados de entrada para uma rede convolucional), acrescida de ajusto fino aos dados de saída, finalizando com a classificação das imagens entre real e sintética/falsa.

Para a melhor organização deste capítulo, descrevemos a abordagem que utilizamos para a organização e análise dos dados (Seção 4.1). Apresentamos o processo para a extração de faces e *pacthes* das imagens e divisão das bases de dados para utilização em nossos experimentos (Seção 4.2). Exploramos abordagens para a utilização de arquiteturas de redes convolucionais, considerando cenários com diferentes taxas de aprendizado, métodos de otimização, configurações de camadas e taxa de congelamento das camadas (Seção 4.3) na tentativa de resolver o problema de detecção de *DeepFake*s. Por fim, com o intuito de melhorar os nossos resultados, também apresentamos propostas de utilização de aumentação de dados (Seção 4.4), diferentes tipos de representação de entrada (Seção 4.5), e funções de custos (Seção 4.6).

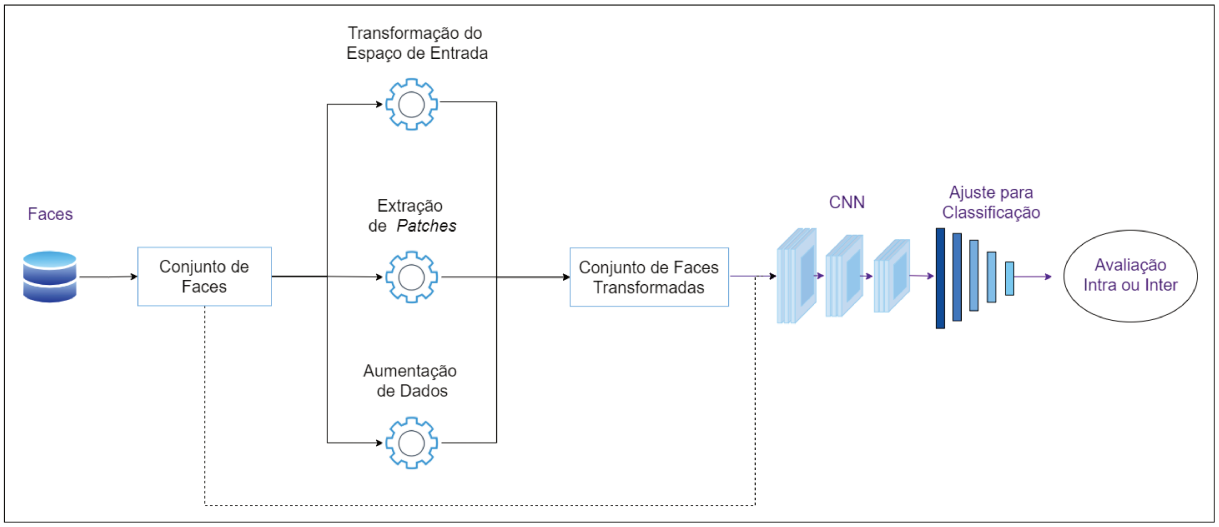


Figura 4.1: Na nossa proposta de trabalho utilizaremos apenas as faces que foram extraídas de cada base de dado. O conjunto total de faces é dividido em subconjuntos que poderão ser utilizados diretamente como entrada para a rede ou passarão por alguma transformação nos dados antes de serem enviado a rede. Em seguida, utilizamos a CNN para extrair características que serão ajustadas por camadas totalmente conectadas fornecendo o resultado para a classificação intra ou inter bases de dados.

4.1 Organização dos Dados

Nessa seção apresentaremos a nossa metodologia para criação na nossa base de dados (Seção 5.1.2), a `Sensitive`, assim como as análises e manipulações necessárias para a utilização da base de dados `DFDC` para o treinamento dos nossos modelos (Seção 4.1.1). Por fim, também apresentamos o método utilizado para divisão dos dados entre treino e validação (Seção 4.2.2).

4.1.1 DeepFake Detection Challenge

Para a utilização do `DFDC`, foram necessários alguns passos adicionais tais como a análise e seleção dos vídeos, conforme apresentado a seguir.

Análise

A disponibilização do `DFDC` trouxe uma evolução em relação às bases de dados anteriores. Neste conjunto de dados, diversos tipos de técnicas foram aplicadas para gerar vídeos com *DeepFake*. Entretanto, diferentemente das outras bases, os autores do `DFDC` não disponibilizaram detalhes sobre os dados, com o intuito de tornar a tarefa de detecção mais desafiadora.

Dito isso, surgiu a necessidade da realizar uma análise preliminar da base para identificar algumas das suas características. Nessa etapa, buscamos obter informações como, a quantidade de faces por vídeo, no caso de haver múltiplas faces e em quais delas foram aplicadas *DeepFake*. Adicionalmente verificamos se havia oclusões faciais capazes de impedir a localização e extração das faces.

Seleção de Vídeos

Conforme apresentado na Seção 5.5, os vídeos sintéticos/falsos da base de dados DFDC podem conter manipulações de face ou áudio. Considerando que não foram disponibilizadas informações que nos permitam separar a abordagem aplicada, foi necessário encontrar um método capaz de nos auxiliar na identificação de quais vídeos possuem alterações na face. Desse modo, evitamos que sejam introduzidas imagens sem *DeepFake* no conjunto de imagens sintéticas/falsas.

Para isso, utilizamos o *SSIM* (Subseção 6.8) para computar a similaridade entre faces reais e sintéticas/falsas, do seguinte modo: para cada vídeo do DFDC rotulado como real (VR), selecionamos os vídeos sintéticos/falsos (VF) relacionados; com o método da Seção 4.2 extraímos todas as faces de VR compondo as faces reais (FR). Utilizando a localização obtida para as FR e extraímos as faces sintéticas/falsas (FF).

Em seguida, para cada par (FR, FF) computamos o *SSIM* médio, que pode ser definido pela Equação 4.1.

$$\overline{SSIM}(FR, FF) = \frac{\sum_{i=1}^{n} \text{SSIM}(FR_i, FF_i)}{n} \tag{4.1}$$

onde n é a quantidade de faces e $F\cdot_i$ representa a i-ésima face do conjunto $F\cdot$ (veja Figura 4.2). Um conjunto FF_i é utilizado para compor a classe sintética/falsa se e somente se $\overline{SSIM}(FF, FR) \leq L$, onde L é um limiar definido empiricamente.

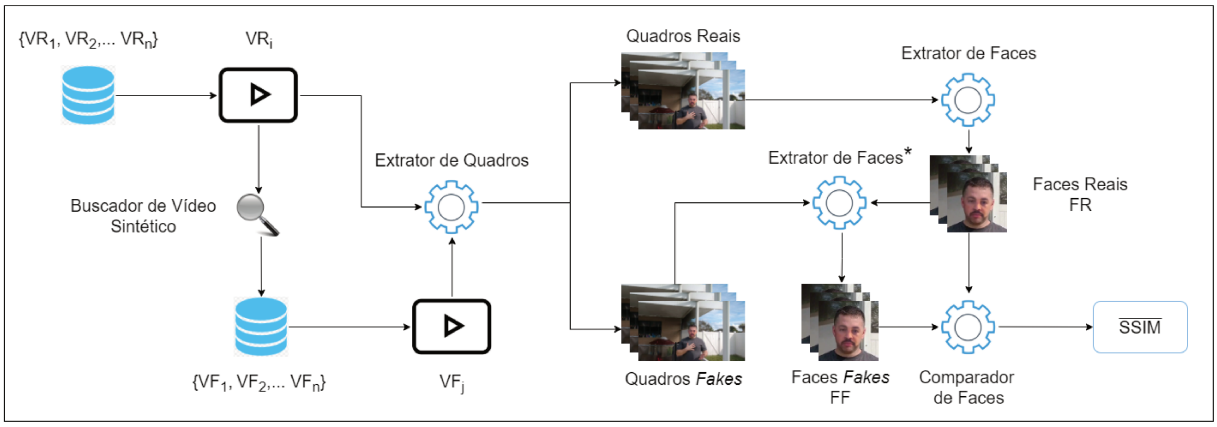


Figura 4.2: Método utilizado para a seleção de vídeos do DFDC. Para cada vídeo real VR_i, buscamos todos os vídeos sintéticos/falsos (VF_j) gerados a partir de VR_i. Para cada par (VR_i, VF_j), são extraídos os quadros e, em seguida, as faces dos quadros reais. A localização obtida em FR_i é utilizada para extrair as FF_i. Por fim, é computado a similaridades entre os dois conjuntos de faces. * Esse extrator utiliza as localizações de faces previamente computadas na extração de FR_i.

4.2 Processamento dos Dados

Uma etapa importante para o desenvolvimento deste trabalho, consiste em preparar os dados para a utilização. Para isto, utilizamos pontos de referência para extrair tanto a face quanto regiões de interesse (Subseção 4.2). E realizamos a divisão dessas imagens após a extração,para que pudessem ser apropriadamente utilizadas em nossos experimentos.

4.2.1 Extração de Faces e *Patches*

Dado um conjunto de vídeos V, o primeiro passo da pesquisa consiste na extração de quadros Q do vídeo à uma taxa de f quadros por segundo. Investigamos diferentes valores para f até encontrarmos o ideal, de tal modo que não haja a repetição de um mesmo quadro e nem percamos informações importantes para a detecção de *DeepFake*. Também utilizamos valores recomendados pelos criadores das bases. Em seguida, utilizamos Q para realizar a extração de faces F.

Para isso, adotamos um modelo de aprendizado profundo (conforme apresentado na Seção 2.2) que encontra uma região retangular na qual esteja contida uma face. Adicionamos um *pad* ξ e υ na largura e altura, respectivamente, para ampliar a região facial e evitar perda de informações próximas à região de borda da face. Tais retângulos, também são utilizados para extrair regiões de interesse (*patches*).

Com o mapeamento dos pontos de referência das faces extraídas, localizamos as regiões da boca, nariz e olhos. Cada região da face é composta por um conjunto de pontos, sendo que a região é determinada pelo retângulo de menor lado que contenha todos esses pontos. Para montar a região dos olhos, utilizamos os pontos referentes aos olhos (esquerdo e direito) assim como os das sobrancelhas (esquerda e direita) e montamos um único *patch* (vide Figura 4.3). Essas regiões são então utilizadas para o treinamento dos modelos de aprendizado.

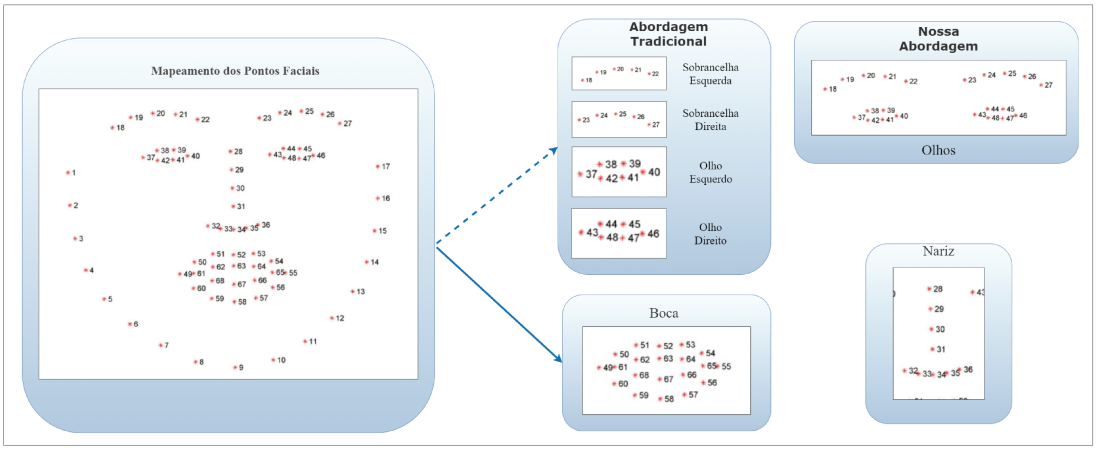


Figura 4.3: Método para extração da e composição da região dos olhos. Conforme descrito no quadro abordagem tradicional, cada região (olhos e sobrancelhas) são extraídas individualmente. Para a nossa abordagem, após a extração unimos essas regiões para compor uma única imagem de olhos. Além disso, ainda utilizaremos a boca e o nariz. Adaptada de Rosebrock [97].

4.2.2 Conjunto de Treinamento e Validação

As bases utilizadas neste trabalho não foram originalmente separadas entre treino e validação. Neste contexto, propomos uma abordagem para dividir os dados baseada no indivíduo, e utilizaremos a proporção de 80% para treinamento e 20% para validação.

Para isso, selecionamos os vídeos de acordo com as identidades. Isto é, todos os vídeos de mesma pessoa devem pertencer ao mesmo conjunto. Caso esta divisão não seja possível,

como no DFDC onde não é fornecida nenhuma identificação dos atores, selecionamos o vídeo e todos os vídeos sintéticos/falsos que foram gerados a partir deste para compor o mesmo conjunto.

4.3 Método Base: Modelo Pré-Treinado

Nosso problema consiste em realizar a classificação de uma imagem entre as classes real e sintéticas/falsas. Em nossa proposta de solução, adotamos duas abordagens. Na primeira, realizamos os experimentos iniciais com classificadores tradicionais, tais como, SVM e Perceptron Multicamadas (*Multi-layer Perceptron*, MLP) [105], extraindo características por meio de CNNs. Já para a segunda, treinamos CNNs com transferência de aprendizado e ajuste fino dos dados.

Para ambas as abordagens, selecionamos arquiteturas de aprendizado profundo consolidadas para problemas de classificação, tais como, MobileNet[43] e ResNet[40], pré-treinadas em outras bases de dados. Essas arquiteturas são particularmente importantes, devido a seu bom desempenho em tarefas gerais de classificação.

A abordagem tradicional, apresentada na Figura 4.4, representa nosso *baseline*. Entretanto, nosso foco consiste no treinamento de novos modelos, explorando diversos aspectos das CNNs e do processo de treinamento. Logo, um dos aspectos que exploramos em todos os nossos modelos é a utilização de transferência de aprendizado.

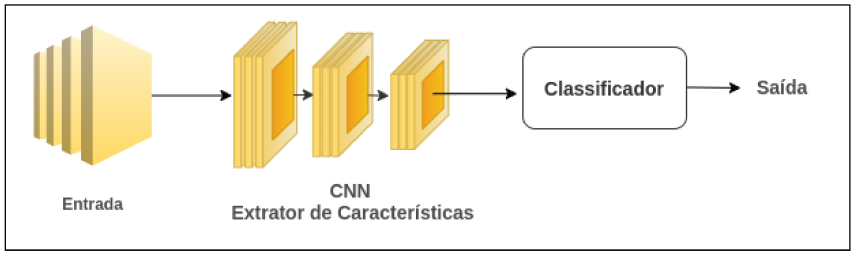


Figura 4.4: Proposta de modelo base. Utilizamos um subconjunto de faces como informação de entrada para uma rede, como a *ResNet*, *MobileNet* ou *Xception*. Em seguida, adicionamos um classificador, que pode ser um SVM/MLP ou algumas camadas totalmente conectadas. Ao final, a saída da rede é uma classificação binária.

Neste caso, adotamos uma rede pré-treinada, isto é, que foi treinada em uma outra base de dados (normalmente com milhões de exemplos) e, então, seus pesos são ajustados para uma nova tarefa de classificação. Além de ajudar a prevenir *overfitting*, a transferência de aprendizado é eficaz, pois muitos conjuntos de dados de imagem compartilham características espaciais de baixo nível que são melhor aprendidas em grandes conjuntos de dados [86]. Neste trabalho, adotamos os pesos das camadas convolucionais treinados no conjunto de dados *ImageNet*[1][18]. Para isto, é necessário definir quais são as últimas camadas da rede[2] a serem ajustadas/adaptadas, processo conhecido por ajuste fino. Nesta

[1] *ImageNet* é um conjunto de dados com mais de 15 milhões de imagens, rotuladas com cerca de 22.000 categorias.

[2] Essas camadas são normalmente chamadas de topo da rede.

etapa, testamos diversas combinações de camadas lineares, *pooling* e regularização (como por exemplo, *batch normalization* e *dropout*).

Também exploramos hiper-parâmetros, como a taxa de aprendizado, taxa de congelamento das camadas, assim como diferentes otimizadores, tais como, Adam[61], Nadam[28], Adamax[61], RMSprop[38] e SGD[115]. Após explorarmos todos esses aspectos da rede, também exploramos alterações nos dados de entrada e técnicas de aumentação de dados, conforme apresentado nas próximas seções.

4.4 Método 1: Aumentação de Dados

Com a premissa de que mais dados de treinamento proporcionam maior representatividade do problema, nessa seção, exploramos o desempenho de técnicas de aumentação de dados — estratégia que permite aumentar a diversidade dos dados disponíveis, sem necessariamente coletar novos dados [42].

As aumentações de dados adotadas são baseadas em modificações de imagens usando operações de transformação geométrica e radiométrica [73, 109]. Neste contexto, selecionamos algumas aumentações como: rotação, espelhamento (vertical e horizontal) e translação que não alteram propriedades estruturais da imagem, além de brilho e ampliação, focando em expor possíveis artefatos. Estas foram aplicadas em tempo real durante o processo de treinamento, ou seja, as técnicas escolhidas são inseridas a cada *batch* do treinamento, definidas randomicamente de acordo com os intervalos dos parâmetros. As aumentações geradas não são salvas em disco, otimizando todo o processo. Uma representação da nossa proposta pode ser vista na Figura 4.5. Adicionalmente, também exploramos diferentes representações de entrada, conforme apresentado na próxima seção.

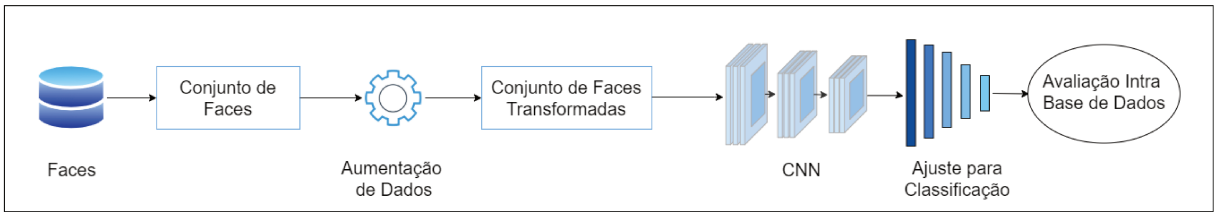


Figura 4.5: Dado o conjunto total de faces, geramos subconjuntos em que aplicamos as aumentações definidas e, com estas novas imagens, extraímos características para a classificação entre real e falsa.

4.5 Método 2: Representações de Entrada

A abordagem padrão para treinar CNNs consiste em utilizar as imagens originais como entrada para a rede de aprendizado, que, por sua vez, realiza a extração de características e, em seguida, faz a classificação. Neste trabalho, também exploramos outras representações, tais como, *patches* (Seção 4.5.1) e transformação do espaço de entrada (Seção 4.5.2). Acreditamos que ao acrescentar novas informações para a rede, podemos identificar características que auxiliarão da detecção de faces manipuladas.

4.5.1 *Patches*

Essa abordagem consiste na extração de *patches* das faces (conforme seção 4.2) e sua utilização para treinamento dos modelos ao invés da região da face como um todo. Acreditamos que treinar uma rede para cada região da face pode ajudar na identificação mais precisa, principalmente, para casos com *DeepFake* foto-realistas (alta qualidade visual) ou entre pessoas parecidas.

Para isso, testamos algumas combinações, como por exemplo, utilizar todos os *patches* (boca, nariz e olho) individualmente em uma CNN, além da própria face e juntar as características no final das redes pré-treinadas. Além disso, também testamos cada *patch* individualmente para verificar o quanto cada um deles contribui no processo. Uma representação da nossa proposta é apresentada na Figura 4.6.

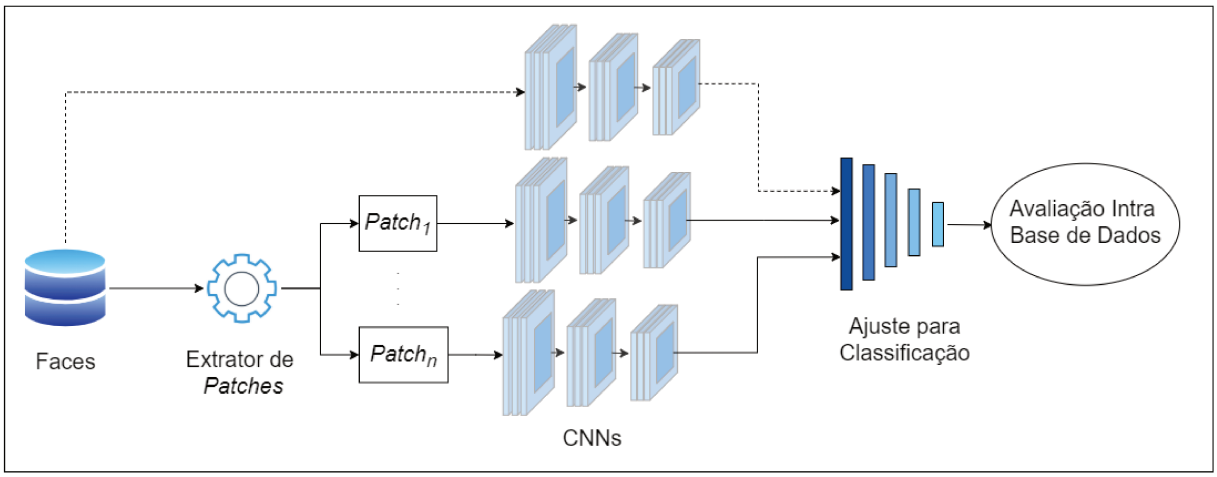


Figura 4.6: Na primeira abordagem, realizamos o treinamento da rede apenas com os *patches*, tanto individuais quanto em conjunto. Já a segunda abordagem, acrescentamos ao treinamento além de cada *patch* a face.

4.5.2 Transformação do Espaço de Entrada

Outra representação de entrada que exploramos no trabalho envolve a transformação da imagem de entrada para outro espaço de representação alternativo. A ideia consiste em extrair características que possam fornecer à rede informações adicionais para o aprendizado. Como exemplo de espaços de entrada transformados, temos iluminantes[94], profundidade[65], reflectância e Albedo[93].

Cada uma dessas técnicas propõe a extração de características específicas das imagens, como, informações de iluminação, cores e profundidade. As imagens sintéticas/falsas podem apresentar diferenças nestas características, quando comparadas com as originais. Acreditamos que a utilização de tais técnicas pode auxiliar na tarefa de detecção de *DeepFake*.

Inicialmente, utilizamos cada mapa individualmente como entrada da rede. Entretanto, nosso objetivo é fornecer os mapas como entrada adicional juntamente com as imagens originais, isto é, aumentarmos a quantidade de canais para a entrada da rede, sendo os três primeiros correspondentes aos canais da imagem original (RGB) e os de-

mais compostos pelos mapas. Nossa proposta é ilustrada na Figura 4.7. Nesta, para cada face do conjunto total de faces, realizamos a sua transformação gerando uma nova representação (por exemplo, mapa de profundidade). Então, dividimos o processo em duas abordagens. Na primeira apenas a nova informação é utilizada para o treinamento, e como necessitamos de três canais para a rede, replicamos o mesmo dado para todos os canais. Na segunda abordagem, associamos a face em RGB. Independente da abordagem, finalizamos aplicando camadas convolucionais para ajustar os dados conforme a entrada esperada pela rede e realizamos a classificação.

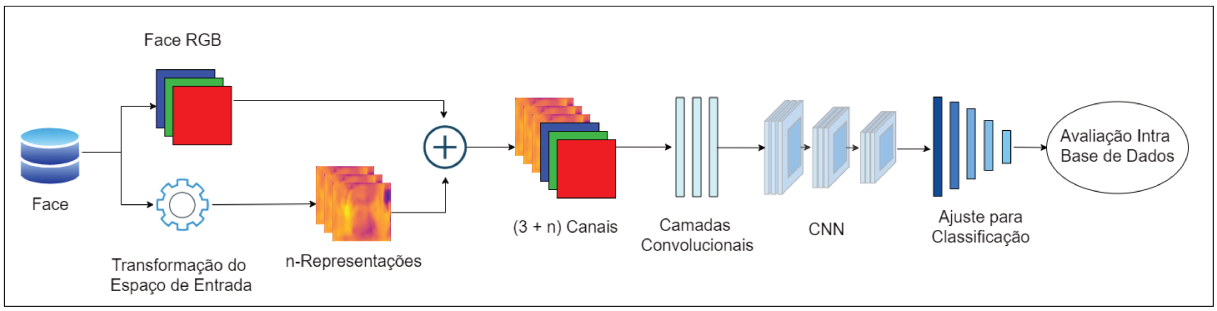


Figura 4.7: Representação do espaço transformado de entrada, em suas duas abordagem.

4.6 Método 3: Funções de Custo

No aprendizado de máquina, utilizamos uma função de custo para avaliar uma solução candidata, que pode ser tanto de minimização quanto de maximização. Quando tentamos minimizar o erro, a função utilizada é chamada de perda [7]. De acordo com a proximidade entre as previsões e os rótulos, obtemos um valor para a função de perda que é minimizado com a utilização de otimizadores [128].

Quando falamos de classificação, uma das funções de perda mais utilizadas é a Entropia Cruzada, definida na Subseção 4.6.1. Entretanto, também utilizamos neste trabalho a Função de Perda Tripla *(Triplet Loss)*, apresentada na Subseção 4.6.2.

4.6.1 Entropia Cruzada

A entropia cruzada é uma das funções de perda usadas em problemas de classificação e mede o desempenho de um modelo cuja saída é um valor de probabilidade entre 0 e 1. Essa perda aumenta à medida que a probabilidade prevista diverge do rótulo real [89].

Na classificação binária, a entropia cruzada pode ser calculada pela Equação 4.2.

$$Perda_{EC} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{4.2}$$

onde y é o rótulo da classe e p é a probabilidade predita pelo classificador. Dependendo do valor de y uma parte da função será descartada e, com isso, as previsões erradas serão penalizadas [89]. No nosso trabalho, $y = 1$ quando as imagens são falsas e $y = 0$ para os casos reais.

4.6.2 Perda Tripla

De acordo com o Schroff et al. [104], a perda tripla é uma função que tem o objetivo de minimizar a distância entre objetos da mesma classe enquanto maximiza a distância com objetos de classes diferentes. Para isso, uma âncora é escolhida junto com uma amostra negativa e uma positiva. Então, minimiza-se a distância entre a âncora e as amostras positivas enquanto se maximiza a distância com as amostras negativas, por meio da Equação 4.3. Na Figura 4.8, apresentamos uma representação deste processo.

$$L(\alpha, P, N) = max(||f(\alpha) - f(P)||^2 - ||f(\alpha) - f(N)||^2 + m, 0) \tag{4.3}$$

onde α é a âncora, P são as amostras positivas, N as negativas e m é a margem.

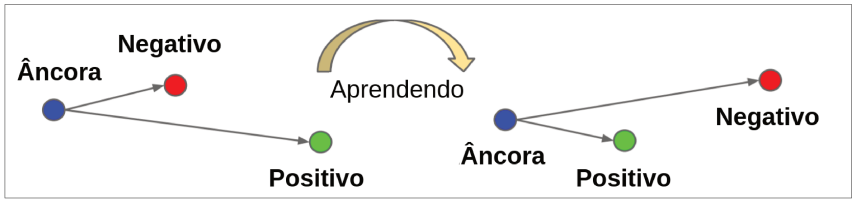


Figura 4.8: Esquema de aprendizado da perda tripla. À esquerda, temos uma tripla que, após o processo de aprendizagem, consegue minimizar a distância entre a âncora e a amostra positiva e maximizar a distância entre a âncora e a negativa. Adaptada de Schroff et al. [104].

As triplas podem ser divididas em: fáceis, semi-difíceis, difíceis e mais difíceis, conforme apresentado na Figura 4.9.

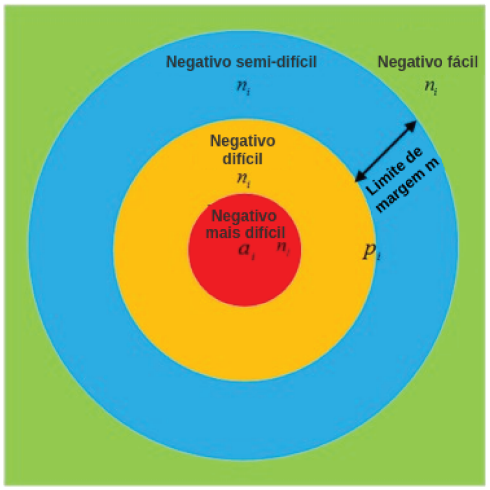


Figura 4.9: Tipos de funções de perda triplas de acordo com a diferença entre âncora e negativo e âncora e positivo. Adaptada de Zhu et al. [138].

Neste trabalho, utilizamos a tripla semi-difícil que emprega o aprendizado online[3] para selecionar uma âncora. Conforme, Schroff et al. [104], esta versão apresenta melhores resultados do que a perda tripla original. Nesta formulação, a amostra negativa tentará

[3]No aprendizado online, a escolha da tripla é feita durante o processo de treinamento.

maximizar a sua distância Euclidiana da âncora, mas ainda será capaz de oferecer uma perda positiva [123], conforme Equação 4.4.

$$d(\alpha_i, P_i) \leq d(\alpha_i, N_i) \leq d(\alpha_i, P_i) + m \tag{4.4}$$

onde m é a margem, α_i é a âncora, P_i a localização da imagem positiva, N_i da negativa e d é a distância entre as imagens [138].

Capítulo 5

Conjuntos de dados

Um dos desafios iniciais deste trabalho foi encontrar conjuntos de dados que representassem o problema que queremos resolver. Para isso, além de termos criado a nossa própria base de dados, nos mantivemos em constante busca por novos conjuntos, à medida que se tornavam disponíveis.

Nas seções seguintes, descrevemos cada base de dados utilizada, suas características e a quantidade de vídeos ou imagens que os compõe. Na Seção 5.7 apresentamos um resumo das principais informações destes conjuntos.

5.1 Sensitive

Esta coleção é uma importante contribuição deste trabalho, e consistiu na criação e futura disponibilização de uma base de dados sintética/falsa, coletada em sites de conteúdo adulto, analisados e devidamente anotados. Descreveremos nas subseções seguintes o processo para criação do `Sensitive` e como o combinamos com a base de dados de imagens reais[1] *VGG_ Faces2* [12].

5.1.1 Conjunto de Imagens Reais

Conforme mencionado anteriormente, escolhemos compor a nossa classe de imagens reais utilizando a base de dados `VGG_Faces2`, sendo necessária para manter a representatividade equivalente entre as classes de imagens reais e sintéticas/falsas. Acreditamos que esta base é suficientemente representativa, composta apenas por imagens com ênfase na face e bastante utilizada em tarefas de reconhecimento facial. Além disso, o `VGGFaces2` apresenta variabilidade de cenários e características físicas dos indivíduos, o que é condizente com o que buscamos.

Conforme descrito por Cao et al. [12], o `VGG_Faces2` foi obtido por meio da pesquisa individual de celebridades no *Google* imagens. Sendo selecionadas, ao todo 9.131 celebridades. E foram consideradas aquelas em que estivesse disponível, no mínimo, 80 imagens por identidade, totalizando 3.31 milhões de imagens.

[1] Consideramos dados reais, aqueles em que não foram aplicados técnicas de alteração da face.

Para garantir um bom conjunto de dados, além da variação de pose, idade, etnia, iluminação e fundo, foram realizadas filtragens semânticas para minimizar a ocorrência de imagens não relacionadas à busca. Adicionalmente, essas imagens passaram por uma etapa de detecção facial para garantir que a cabeça do indivíduo estaria visível. Ao final destas etapas, foram removidas imagens com sobreposição de rostos (ou seja, quando há a obstrução parcial da face), imagens duplicadas e não relacionadas ao indivíduo.

Por fim, o conjunto de dados é dividido em duas partes: treinamento com 8.631 celebridades e teste com 500. Em nosso trabalho, utilizamos apenas o conjunto de testes que representa 164.406 imagens, proporcional à quantidade de imagens sintéticas/falsas que utilizaremos.

5.1.2 Conjunto de Imagens Sintéticas/Falsas

Para compor a classe sintética/falsa do conjunto de dados e após o acompanhamento de fóruns, percebemos que a principal fonte de vídeos sintéticos/falsos são sites de conteúdo adulto. Com isso, fizemos o levantamento de uma lista com os principais sites especializados neste tipo de conteúdo com *DeepFake*s e iniciamos a busca por vídeos rotulados com *DeepFake*s e coleta de vídeos. Inicialmente, coletamos todos os vídeos com *DeepFake* disponíveis nos sites, totalizando 420 vídeos.

Em seguida, realizamos algumas verificações a fim de garantir a qualidade da base. Selecionamos manualmente alguns vídeos, de modo a eliminar os que continham mais de um indivíduo. Nos asseguramos que os vídeos selecionados continham *DeepFake*s, por meio da escolha de sites que disponibilizavam apenas vídeos em que a técnica é aplicada. Além disso, a face da celebridade inserida, intitulava o vídeo. Com isso, removemos 120 vídeos com baixa qualidade ou múltiplas faces. Os demais vídeos foram extraídos a uma taxa de 4 FPS, e, das imagens resultantes, foram extraídas as faces (conforme Seção 6.1.3), totalizando 118.000 faces sintéticas/falsas.

O processo de construção desta base pode ser descrito pelos passos a seguir.

Levantamento: Inicialmente, realizamos um levantamento de onde podem ser encontrados vídeos ou imagens com *DeepFake*s. Isso foi realizado através do monitoramento de fóruns que continham referência a esse tipo de conteúdo em redes sociais como o *Twitter* e o *Reddit*.

Coleta: Em seguida, realizamos a coleta desses vídeos para compor a nossa base.

Processamento: Para esta etapa, fizemos o processamento dos dados coletados a fim de verificar a qualidade dos vídeos. Nesse momento, removemos os vídeos que apresentaram características indesejadas, tais como, múltiplas faces.

Extração de imagens: Após o processamento, fizemos a extração de quadros dos vídeos selecionados, de modo a gerar um conjunto de imagens.

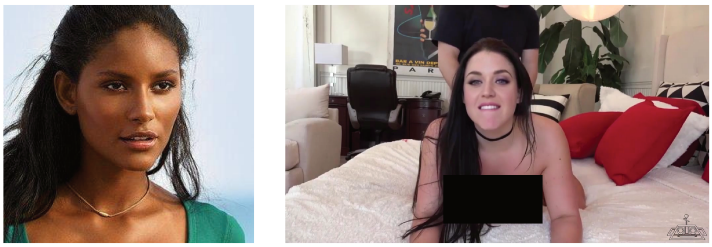
Composição do conjunto final: Compomos a nossa base final `Sensitive` associando as imagens reais (5.1.1) e as imagens sintéticas/falsas coletadas por nós.

Extração de Faces: Para cada imagem da base de dados (real e sintética/falsa) utilizamos algoritmos de detecção facial para selecionar apenas as faces (conforme Seção 4.2).

Rotulação: Por fim, a rotulação consistiu na separação das imagens em duas classes: real, contendo imagens do *VGG_ Faces2* e sintética/falsa.

Ressaltamos que ao utilizarmos estes vídeos, obtivemos variabilidade em relação à qualidade do vídeo, iluminação, cenário, posição da cabeça e tipo da técnica aplicada. Entretanto, podem ocorrer oclusões parciais da face em alguns vídeos, o que pode facilitar a tarefa de classificação, tendo em vista que nestes momentos a máscara de *DeepFake* distorce na imagem. Além disso, por ser composto apenas de vídeos femininos estamos suscetíveis ao enviesamento da classificação.

Na Figura 5.1, apresentamos exemplos das bases utilizadas para compor o `Sensitive`.



(a) (b)

Figura 5.1: Imagens da base de dados `Sensitive`. (a) Imagem real do `VGG_Faces2` e (b) Imagem sintética/falsa do `Sensitive`.

5.2 Notre Dame Synthetic Face - NDSF

Este conjunto de dados apresenta imagens de rostos de identidades sintéticas ou seja, identidades criadas virtualmente com diferentes formas e resoluções faciais. O utilizamos para incrementar o conteúdo sintético/falso dos conjuntos de dados.

As faces sintéticas foram criadas a partir da combinação de rostos reais com aparência similar. Para isto, o método de síntese de imagens de rosto utiliza uma imagem frontal do rosto e realiza a triangulação da região facial a partir dos pontos de referência. Em seguida, os triângulos são substituídos pelos triângulos correspondentes de outras imagens de rosto, que serão mesclados para obter uma textura de rosto [6].

Apresentamos na Figura 5.2 exemplos das imagens sintéticas deste conjunto de dados.

Figura 5.2: Imagens da base de dados `Notre Dame Synthetic Face` nas três posições que utilizamos neste trabalho.

5.3 FullTIMIT

O conjuntos de dados `FullTIMIT` é composto por vídeos reais da base `VidTIMIT` [102] e vídeos sintéticos/falsos do `TIMIT` Pavel Korshunov [90], que foram gerados a partir da base `VidTIMIT`.

O `VidTIMIT` é composto por vídeos de 43 pessoas recitando dez frases curtas e realizando uma sequência de rotação da cabeça. Todas as gravações foram realizadas em um ambiente de escritório usando uma câmera de vídeo digital, e foram armazenadas em formato de imagens *jpeg* com resolução de 512×384 pixels.

Já o `TIMIT` contém vídeos sintéticos/falsos gerados a partir de um software de código aberto baseado em GANs. A abordagem utilizada por Pavel Korshunov [90] considera a seleção de 16 pares de pessoas com aparência semelhante para a produção de vídeos sintéticos/falso. Sendo estes gerados com as resoluções: baixa (*low quality*, LQ) com tamanho de entrada / saída de 64×64, e alta qualidade (*high quality*, HQ) com tamanho de 128×128. Ao fim deste processo, são gerados 640 vídeos (320 vídeos para cada resolução).

O método para geração de vídeos sintéticos/falsos é dividido de acordo com as resoluções dos vídeos. Para os vídeos LQ, uma face foi extraída a cada quadro. Em seguida, uma máscara facial foi detectada usando um algoritmo de segmentação baseado em CNN (ver Seção 2.3) e foi combinada com a face no vídeo de destino. Já para o HQ, foi realizado o alinhamento dos pontos fiduciais entre a face gerada e a original no vídeo de destino. Sendo aplicada a normalização do histograma para mesclar a face gerada a cena de modo a ajustar as condições de iluminação [90].

Na Figura 5.3 as imagens (a) e (b) representam o ator e o alvo, respectivamente. E as imagens (c) e (d) são os resultados da sobreposição do ator no alvo.

(a) (b) (c) (d)

Figura 5.3: Imagens da base de dados `FullTIMIT`. (a) Ator e (b) Alvo obtidos do `VidTimit`. Imagem sintética/falsa do `TIMIT` em alta (c) e baixa (d) resolução.

5.4 Face Forensics Plus Plus, FFpp

Esta base de dados foi desenvolvida a partir de 1.000 vídeos reais, manualmente analisados, a fim de garantir uma seleção sem oclusões faciais.

A partir dos vídeos reais, foram produzidos vídeos *fake*, por meio de métodos baseados em Computação Gráfica e Aprendizado de Máquina. Para o primeiro método, foram desenvolvidos o *Face2Face*, que consiste na re-encenação facial em tempo real e o *FaceSwap*, baseado no aplicativo que troca o rosto de alvo pelo rosto de um ator. A outra

abordagem produz *DeepFake* a partir do aprendizado profundo utilizado para reconhecer e trocar rostos e texturas [33, 64, 124, 125].

Os vídeos foram gerados com compressões diferentes: vídeos brutos, sem compressão (C0), HQ com taxa de compressão de 23 (C23) e LQ com quantização de 40 (C40). O intuito foi simular o processamento de vídeo de redes sociais.

Como resultado, foi produzido um conjunto de dados composto por 4000 vídeos sintéticos/falsos. Na Figura 5.4, mostramos imagens divididas pela técnica e taxa de compressão aplicada.

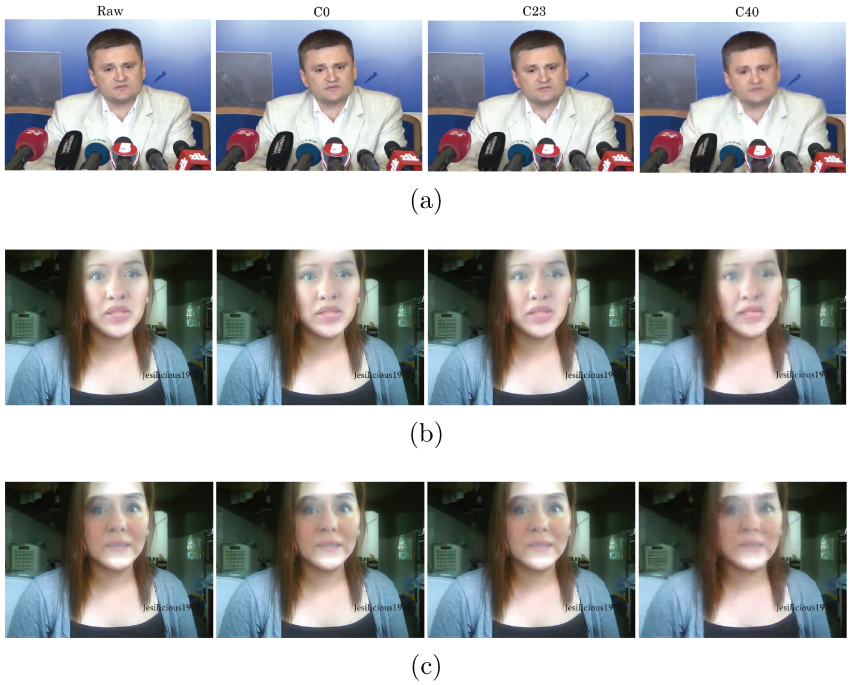


Figura 5.4: Imagens da base de dados `Face Forensics Plus Plus`. (a) Imagem real, (b) Imagem sintética/falsa gerada a partir da técnica de *FaceSwap* e (c) Imagem sintética/falsa utilizando a técnica de *DeepFake*.

5.5 DeepFake Detection Chalange

O conjunto de dados `DeepFake Detection Chalange` (DFDC) é composto por 119.347 vídeos no formato mp4. A base contém pessoas de diferentes origens — caucasianos, afro-americanos e asiáticos —, em cenários distintos e com condições variadas de iluminação, e diversos tipos de movimentação da cabeça.

As trocas de rostos foram realizadas entre indivíduos com aparências semelhantes, considerando atributos faciais como tom de pele, pêlos faciais, uso de óculos, entre outros. O vídeo sintético/falso foi gerado utilizando duas identidades, sendo a primeira pessoa o alvo e, em seguida, trocando o alvo para a segunda identidade. Assim, ambas faces foram utilizadas para gerar novas identidades.

Cada vídeo completo tem cerca de 30 segundos, que foram divididos em clipes com 10 segundos cada. Todos os clipes estão rotulados de modo que cada vídeo sintético/falso possa ser relacionado ao vídeo real correspondente. Além disso, nesta base de dados,

também há vídeos com modificações apenas de áudio, ou seja, o áudio foi manipulado de modo que não corresponda àquela pessoa ou que partes do áudio original tenham sido alteradas. Esta informação não foi rotulada, assim, é necessário identificar e separar estes vídeos [26].

A Figura 5.5 apresenta a variação de condições a que a base de dados foi exposta.

(a) (b)

Figura 5.5: Imagens da base de dados `DFDC`. (a) Imagem real e (b) Imagem sintética/falsa.

5.6 DeepFake Detection

O `DeepFake Detection` (DFD) [30] é uma base de dados desenvolvida em colaboração entre Google e Jigsaw[2][46], sendo incorporada ao *benchmark* da *University Federico II of Naples'* com parceria da *Technical University of Munich*.

Para a sua produção, foram gravados vídeos com alguns atores e, a partir destes, os vídeos *DeepFake*s foram produzidos utilizando métodos publicamente disponíveis, gerando milhares de vídeos com *DeepFake*. A base está disponível gratuitamente para a comunidade de acadêmica com o intuito de auxiliar o desenvolvimento de métodos para a detecção de vídeos sintéticos/falsos.

Esta base de dados contém mais de 3.000 vídeos sintéticos/falsos reproduzidos dos vídeos originais de 28 atores. Para rotular os dados, cada vídeo foi identificado por meio de um número único por ator seguido do nome da cena. Já para os vídeos sintéticos/falsos, é utilizado o número do ator alvo seguido do número do ator de origem, adicionado também o nome da cena e um número identificador de 8 caracteres. Apresentamos, na Figura 5.6, um exemplo para esta base de dados.

(a) (b) (c)

Figura 5.6: Imagens da base de dados `DFD - Google`. (a) Imagem real ator, (b) alvo e (c) Imagem sintética/falsa.

[2]Jigsaw é uma unidade do Google que prevê e confronta ameaças emergentes, por meio de pesquisas e tecnologias, que tem como objetivo manter o mundo seguro.

Utilizaremos esta base de dados para realização de teste cego ao final do desenvolvimento dos métodos, que será descrito na Subseção 6.9.

5.7 Resumo dos Conjuntos de Dados

O `FullTIMIT` foi primeiro conjunto de dados que obtivemos, sendo composto por vídeos falsos e reais, divididos entre alta (HQ) e baixa (LQ) resolução. Entretanto, esse conjunto de dados é bastante limitado e não apresenta a variabilidade necessária para o treinamento de uma solução de aprendizado de máquina.

Por essa razão, construímos e rotulamos o nosso próprio conjuntos de dados, nomeado `Sensitive`. Para tanto, utilizamos a base de dados `VGG_Faces2` como sendo a classe real, e para os casos falsos, coletamos vídeos obtidos através de buscas em sites com vídeos rotulados como sintéticos/falsos.

Para incrementar a nossa base de dados de imagens sintéticas/falsas, utilizamos o `Notre Dame Synthetic Face`, base composta apenas por faces sinteticamente geradas.Em Janeiro de 2019, foi lançado o `Face Forensics Plus Plus`, base de dados contendo vídeos coletados no *YouTube*, com diferentes resoluções, dividido pela técnica aplicada na geração do video sintético/falso, sendo: *faceswap*, *deepfake*, ou *face2face*. Ainda em 2019, O Google em parceria com algumas universidades desenvolveram uma base em que atores foram filmados em diferentes cenas, que foram combinadas para produzir uma classe sintética/falsa. Essa base foi incorporada ao projeto do `FFpp`.

Por último, no início de 2020, foi disponibilizado para o desafio `Deep Fake Detection Challenge` (DFDC) uma base de dados produzida em parceria com o *Facebook* e a *Microsoft*.

Na Tabela 5.1, disponibilizamos um resumo dos conjuntos de dados utilizados neste trabalho. Características detalhadas de cada conjunto de dados estão descritas nas próximas subseções.

Dataset	**Real**	**Sintético/Falso**	**Lançamento**
FullTIMIT	320	640	12-2018
Sensitive	164.406*	300	12-2018**
NDSF	-	8.381*	11-2018
FFpp	1.000	1.000	01-2019
DFD - Google	363	3.068	09-2019
DFDC	1.131	4.113	12-2019

Tabela 5.1: Resumo das principais informações das bases de dados. * Este conjunto de dados contém apenas imagens. ** Esta data refere-se à sua criação.

As bases de dados utilizadas neste trabalho foram disponibilizadas à comunidade para auxiliar no desenvolvimento de técnicas para detectar vídeos com rostos manipulados, mediante solicitação de acesso e aceite dos termos de serviço para utilização dos mesmos. Apenas para o conjunto `Sensitive`, em que utilizamos a `VGG_Faces2` não foi necessária solicitação para acesso e uso.

Capítulo 6

Experimentos e Resultados

Iniciamos este capítulo detalhando como preparamos os dados para utilização nos experimentos, quais os pré-processamentos realizados para todas as bases de dados, tais como, extração de quadros e faces, e divisão entre treino e validação (6.1). Em seguida, na Seção 6.2, apresentamos as métricas utilizadas para avaliação dos resultados e os experimentos realizados para detecção de *DeepFake*s, iniciando pela Seção 6.3 na qual mostramos experimentos utilizando técnicas tradicionais, seguindo para a aplicação de aumentação nos dados (Seção 6.4), diferentes representações de entrada (Seção 6.5) e utilizando perda tripla (Seção 6.6), conforme apresentado na Figura 6.1. Com base nos resultados obtidos, na Seção 6.7, há uma reanálise dos dados do `DFDC` em busca de melhores resultados. Na Seção 6.8, comparamos e analisamos todos os resultados obtidos. Descrevemos o resultado do nosso melhor modelo aplicado em um teste cego e comparamos nossos resultados com a literatura (Seção 6.9). E finalizamos na Seção 6.10, com as considerações finais sobre este capítulo.

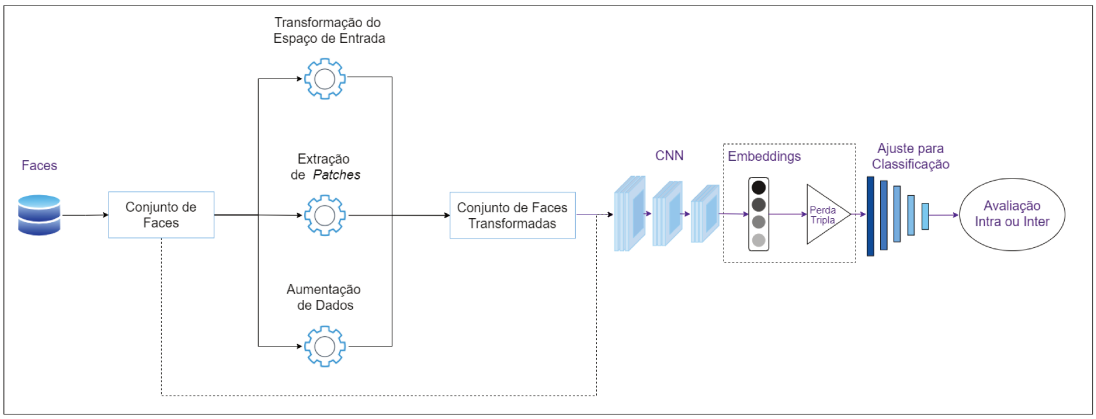


Figura 6.1: Apresentamos a visão geral de todos os experimentos deste capítulo. Partimos das faces extraídas para todos os experimentos. No método base, selecionamos um conjunto de faces que serão utilizadas no treinamento de uma CNN que fará a classificação intra base de dados (linha tracejada). Também a partir de um conjunto de faces, realizamos os experimentos com os demais métodos individualmente, que produzem um novo conjunto com as faces transformadas, que então são utilizadas no treinamento da CNN. Para o método com perda tripla, adicionamos uma camada extra para a rede antes de realizar a classificação final. Por fim, a avaliação inter base de dados segue a mesma configuração do método base.

6.1 Preparação dos Dados

A maioria das informações referentes as bases de dados foram fornecidas pelos seus autores, que mantém uma documentação com as características básicas, método utilizado na geração, resolução, estatísticas relativas a raça e gênero dos indivíduos, tipos de ilumina-ção, entre outros. Além dessas informações, escolhemos aleatoriamente amostras desses vídeos para verificar visualmente algumas características como a qualidade da *DeepFake* aplicada, a taxa de quadros por segundo em que cada vídeo foi gerado e se havia alguma característica que pudesse comprometer o aprendizado dos algoritmos, como imagens com múltiplos indivíduos em que a máscara *DeepFake*estivesse aplicada a apenas um deles.

Entretanto, para o `DFDC`, foi necessário realizar uma análise mais completa, visto que algumas informações específicas foram omitidas devido ao desafio. Na subseção a seguir, detalhamos quais análises realizamos. Em seguida, os vídeos foram convertidos em imagens (Subseção 6.1.2) e as faces extraídas (Subseção 6.1.3).

6.1.1 Análise do `DFDC`

Tendo em vista que o `DFDC` é uma base que não contém muitas informações quanto aos vídeos gerados, realizamos um etapa de análise. Iniciamos utilizando o *FFmpeg* [120] para identificar o tempo de cada vídeo e a taxa de quadros por segundo (*frames per second*, FPS) em que foram gerados. Todos os vídeos contém cerca de 10 segundos de duração e foram gerados a taxas que variam de 8 a 30 FPS.

A partir dessas informações, realizamos diversas análises, sendo a verificação de múltiplas faces a primeira delas. Identificamos que alguns vídeos apresentam mais de uma face, conforme apresentamos na Figura 6.2. No total, foram 1.986 vídeos reais e 11.174 vídeos sintéticos/falsos, sendo estes desconsiderados para o treinamento, pois apesar de a *DeepFake* ter sido realizada em apenas uma pessoa de cada vez, todas faces presente podem ser alteradas em momentos diferentes no decorrer do mesmo vídeo, o que prejudicaria a qualidade da base de dados.

Figura 6.2: Múltiplas faces em um quadro extraídas de um vídeo real do `DFDC`. A ferramenta de extração facial que usamos é capaz de localizar faces em retratos, como os presentes ao fundo desta imagem.

Em seguida, analisamos vídeos onde nenhuma face foi localizada, e identificamos que a abordagem de detecção de faces adotadas não conseguia detectar as faces em algumas

condições, como, por exemplo, pessoas negras ou em ambientes com pouca iluminação. Na Figura 6.3, apresentamos exemplos de alguns destes vídeos. No total, não foram localizadas faces em 91 vídeos reais e, em todos os vídeos sintéticos/falsos derivados desses reais, isto é, 871 vídeos.

Figura 6.3: Imagens em que não foram detectadas faces.

Após a remoção desses vídeos, seguimos para o processo de identificação de vídeos sintéticos/falsos contendo apenas alteração facial (conforme descrito na Seção 5.5), dado que a informação de quais vídeos possuem cada tipo de alteração não foi fornecida. Dito isto, foi necessário remover da base de dados todos os vídeos sintéticos/falsos em que havia a possibilidade de não apresentarem alteração na face.

Para tanto, utilizamos o SSIM, conforme descrito na Seção 6.2.3. Dado um par de vídeos (V_R, V_F), computamos o $\overline{SSIM}$ entre todas as faces sintéticas/falsas (FF) em relação às faces reais correspondentes (FR). Com isso, identificamos que o menor $\overline{SSIM}$ calculado foi de 0.35, o que representa vídeos com menor qualidade da máscara sintética/falsa, como pode ser visualizado na Figura 6.4 (a). Para o limite superior, após diversos experimentos, selecionamos $L = 0.96$, ou seja, vídeos que apresentassem valores de $\overline{SSIM}$ superiores ao definido em L, foram considerados como vídeos sem manipulação na face. Na Figura 6.4 (b), apresentamos uma das faces extraídas com maior $\overline{SSIM}$ dentro do conjunto de vídeos selecionados.

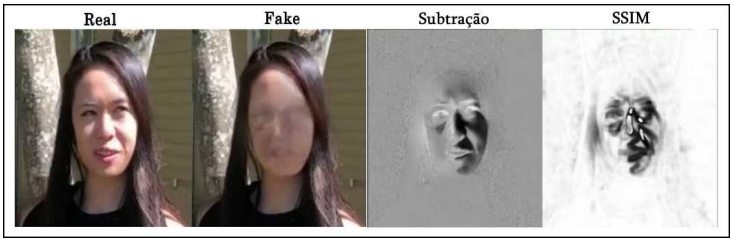
Após a remoção desses vídeos, obtivemos 17.077 vídeos reais e 41.778 sintéticos/falsos. O próximo passo consistiu na extração dos quadros de interesse e faces, como apresentamos nas subseções a seguir.

6.1.2 Extração de Quadros

Em nossos experimentos, trabalhamos apenas com imagens e, considerando que algumas das bases de dados escolhidas contém vídeos, foi necessário realizar a extração dos seus quadros.

Iniciamos utilizando a ferramenta *FFMpeg*, mas essa ferramenta se demonstrou lenta dada a quantidade de vídeos que analizamos. Então, passamos a utilizar a *OpenCV*[1][121].

[1]Biblioteca de software de visão computacional e aprendizado de máquina de código aberto.



(a)

(b)

Figura 6.4: Análise das diferenças entre a imagem real e sua correspondente sintética/falsa. Apresentamos a imagem real, a sintética/falsa correspondente, a diferença entre elas em tons de cinza e o $SSIM$. Na imagem (a) o $\overline{SSIM}$ é de 35, enquanto em (b) este valor corresponde a 99.

Apesar da *OpenCV* utilizar *FFMpeg* em nível de implementação, isso é feito de modo mais otimizado. Essa abordagem nos permitiu decodificar os quadros que seriam de fato utilizados, o que reduziu o tempo de extração.

Para os experimentos com o `DFDC`, extraímos os quadros à uma taxa de 5 FPS (versão 1), a 2 FPS (versão 2) — ambas considerando *SSIM* máximo de 0.97 — e, novamente, a 2 FPS porém com *SSIM* máximo de 0.96 (versão 3). Já para as demais bases, extraímos quadros a uma taxa de 4 FPS, conforme recomendação dada por Pavel Korshunov [90].

Com os vídeos convertidos em imagens, iniciamos o processo de detecção e extração facial, apresentado na subseção 6.1.3.

6.1.3 Extração de Faces e *Patches*

Para a primeira base de dados utilizada nesse trabalho, `FullTIMIT`, utilizamos a técnica de extração baseada na *HaarCascade*, disponibilizada pela *OpenCV*. Esse técnica foi capaz de extrair as faces com sucesso, porém ressaltamos que as imagens eram frontais e sem oclusões. Entretanto, na construção da base de dados `Sensitive`, nos deparamos com um cenário novo em que tínhamos faces em diversas posições e oclusões e, nestes casos, a técnica não se mostrou eficaz, extraindo regiões fora da área de interesse.

Com isso, surgiu a necessidade de encontrar um detector de faces mais robusto. Para tanto, utilizamos duas abordagens. A primeira consistiu na utilização da ferramenta online *Face++*[2][119], entretanto, encontramos limitações para o seu uso, tais como, limite de armazenamento e de solicitações por segundo (*Query Per Second*), além da ferramenta ser de código fechado e não gratuita. Na segunda abordagem, fizemos uma nova

[2]Uma plataforma que oferece tecnologias de visão computacional com suporte para tecnologias de detecção, identificação, reconhecimento e análise de corpo ou imagem, baseado em aprendizado profundo.

implementação utilizando aprendizado profundo, por meio do algoritmo de detecção de objetos de margem máxima (*max-margin object-detection algorithm*), disponibilizado pela Dlib[60]. Como as extrações ocorreram corretamente, adotamos esse método para realizar a localização e extração de faces em todas as demais bases.

Baseada nas faces extraídas, realizamos a extração de *patches* específicos (boca, olhos com sobrancelhas e nariz) utilizando pontos de referência (conforme Figura 6.5), também baseado na implementação disponibilizada pela *Dlib* (Seção 2.2.1). Finalizada as extrações, o próximo passo consistiu na definição dos protocolos de treinamento e validação, conforme seção seguinte.

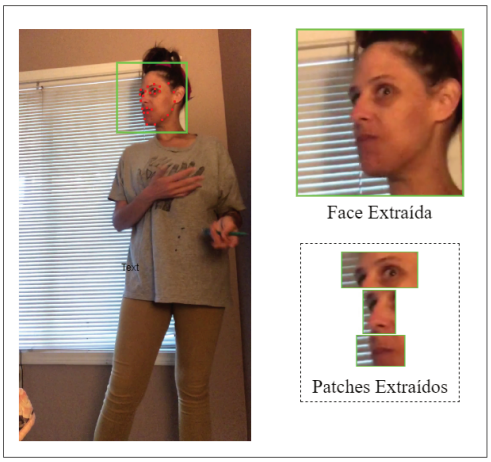


Figura 6.5: Realizamos a detecção da face utilizando os pontos de referência e delimitamos a área final da face. Para a extração dos *patches*, utilizamos a face já extraída para localizar as regiões dos olhos, nariz e boca.

6.1.4 Divisão entre Treino e Validação

Realizamos a divisão dos dados de treinamento e validação. Conforme apresentado na Seção 4.2.2, detalhamos nossa abordagem para a divisão considerando cada base de dados.

Iniciamos com a base `FullTIMIT`, em que realizamos a divisão com base na classe real, onde selecionamos 20% dos indivíduos para a validação e os demais para treino. Já para a classe sintética/falsa, selecionamos os mesmos indivíduos escolhidos para a classe real, de modo a formar um par real-sintético/falso.

Para o `Sensitive`, compomos a classe real com base no arquivo de metadados disponibilizado [12]. Selecionamos 20% dos indivíduos para o conjunto de validação e os restantes para o treinamento. Já para a classe sintética/falsa, escolhemos entre os nomes das celebridades (utilizados como rótulo) também 20% dos vídeos para a validação e o restante foi utilizado para o treino.

Já a classe real do `FFpp` contém um mapeamento para rotulação em relação ao número identificador do vídeo no *YouTube* e uma sequência numérica. Com base nisso, fizemos a divisão dos vídeos reais no padrão 80% treino e 20% validação e utilizamos os mesmos identificadores para encontrar o par de reais utilizados na geração de cada vídeo sintético/falso. Então, selecionamos esses pares de alvo-ator contendo *DeepFake* (1.000 vídeos

em 4 resoluções diferentes totalizando 4.000, conforme Seção 5.4) e *FaceSwap* (também com 4.000 vídeos), separando-os equivalentemente.

Para o DFDC, após a análise realizada na Seção 6.1.1 identificamos que não há nenhuma informação que nos garanta a correta divisão da base por indivíduo. Assim, testamos três abordagens de divisão para os dados. A primeira (versão 1) consiste em aleatoriamente dividir 80% das faces para o treino e 20% para validação. Na segunda abordagem (versão 2), mantivemos a divisão dos dados conforme disponibilizado pelo desafio, ou seja, 50 pastas contendo os vídeos. Então, separamos aproximadamente 20% dessas pastas para a validação e as demais para treino. Para a última abordagem (versão 3), decidimos ordenar os vídeos reais de acordo com seus nomes e selecionamos, em um intervalo uniforme entre 0 e 17.076 (quantidade de vídeos da classe real), 20% das faces para a validação. E, para a sintética/falsa, selecionamos os vídeos gerados a partir dos reais escolhidos, de modo a garantir que os vídeos com as mesmas características, ou seja, condições de iluminação ou cenário, estivessem no mesmo conjunto, isto é, no treino ou na validação, mas não em ambos.

Após realizada essa divisão, definimos quais métricas de avaliação seriam consideradas e iniciamos os nossos experimentos base, conforme descrito na Seção 6.3.

6.2 Métricas

A análise de resultados em aprendizado de máquina é um passo essencial para a tomada de decisões. Para isso, selecionamos diversas métricas utilizadas na literatura tanto para tarefas de classificação, quando para visualização da separação entre a classe real e sintética/falsa.

Para a classificação, utilizamos três métricas. A acurácia durante o treinamento dos modelos e análise da validação. O F_1-*score* utilizado para reportar os resultados no conjunto de validação.E, por fim, a AUC que será utilizada, principalmente, para comparar nossos resultados com a literatura.

Para verificar a separação entre as classes (quando utilizamos a perda tripla), além de utilizar o próprio valor da função de perda, utilizamos a representação visual *UMAP* (Subseção 6.2.2). Um exemplo dessa visualização para a perda tripla é apresentado na Figura 6.6. Além disso, também explicamos o SSIM, que foi utilizado para nos auxiliar na seleção de vídeos do DFDC, conforme apresentado na Seção 6.1.1.

6.2.1 Métricas de Avaliação

A avaliação de algoritmos de aprendizado de máquina é uma parte essencial para o treinamento dos modelos. Os tipos de métricas de avaliação devem ser selecionados de acordo com a tarefa a ser realizada, sendo comumente utilizada a acurácia para medir o desempenho dos modelos de classificação [77].

Nesta seção, apresentamos algumas métricas bastante utilizadas em tarefas de classificação, as quais também foram utilizadas neste trabalho. Dentre elas temos, curva característica de operação do receptor (*receiver operating characteristic*, ROC), área sob

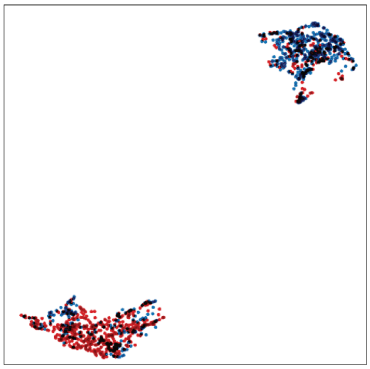


Figura 6.6: Representação UMAP da perda tripla aplicada a alguns dados do conjunto `DFDC`. Os pontos em azul representam os exemplos sintéticos/falsos e, os em vermelho, exemplos reais.

a curva ROC (*area under the ROC curve*, AUC), acurácia, *f1-Score*, precisão e revocação [77]. Também utilizamos o Índice de Similaridade Estrutural (*structural similarity index measure*, SSIM) para comparar faces reais e sintéticas/falsificadas.

Área Sob a Curva

A área sob a curva é usada para problemas de classificação e é calculada sob a curva *ROC*, conforme Figura 6.7 . A AUC equivale à probabilidade de ser classificado um exemplo positivo mais alto do que um exemplo negativo.

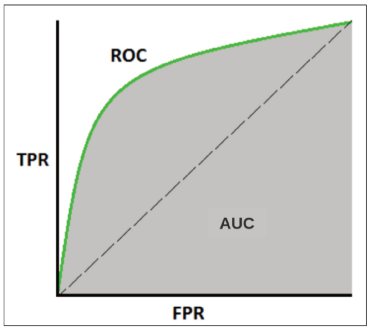


Figura 6.7: Representação da AUC em um gráfico da taxa de verdadeiro e falso positivos. Adaptada de Minaee [76].

Antes de calcular a AUC, é necessário apresentar os termos básicos definidos pelas equações a seguir.

Taxa Verdadeiro Positivo (*True Positive Rate*, TPR): é definida pela Equação 6.1.

$$TPR = \frac{TP}{(FN + TP)} \tag{6.1}$$

onde TP corresponde aos dados positivos que são corretamente considerados positivos, FN são os dados positivos classificados como negativos. Logo, TPR é a relação entre os dados classificados corretamente como positivos e a soma entre todos os corretamente classificados.

Taxa Verdadeiro Negativo (*True Negative Rate*, TNR): é definido pela Equa-

ção 6.2.

$$TNR = \frac{TN}{(TN + FP)} \tag{6.2}$$

onde TNR é a taxa de verdadeiros negativos correspondente à proporção de dados negativos que são corretamente considerados negativos, TN, em relação a todos os pontos de dados corretamente classificados como negativos e os falsos positivos, FP.

Taxa de Falso Positivo (*False Positive Rate*, FPR): definido pela Equação 6.3.

$$FPR = \frac{FP}{TN + FP} \tag{6.3}$$

onde FPR é a taxa de falso positivo, que considera a relação entre os falsos positivos e o conjunto de negativos corretamente classificados e os positivos erroneamente classificados.

Podemos definir AUC como a área sob a curva do gráfico com as taxas de falso positivo e verdadeiro positivo no intervalo entre [0, 1], sendo o desempenho mensurado pelo maior valor de verdadeiro positivo.

Acurácia

Acurácia (*accuracy*, ACC) é a proporção entre o número de previsões corretas e o número total de amostras de entrada, conforme definido na Equação 6.4.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.4}$$

É importante ressaltar que a acurácia não proporciona resultados fidedignos quando utilizada em bases de dados desbalanceadas, ou seja, em que uma classe é maior do que outra [24]. Por isso, ao utilizar essa métrica, selecionamos um subconjunto com a mesma quantidade de exemplos tanto reais quanto falsos.

F_1-Score

F_1-*score* é a média harmônica entre a precisão e a revocação. Com a F_1, é possível precisar a robustez do classificador. Quanto maior o F_1, melhor o desempenho do modelo. Para calcular o F_1, é necessário conhecer a precisão e a revocação.

A precisão é calculada pela Equação 6.5 enquanto a revocação é dada pela Equação 6.6.

$$precisão = \frac{TP}{TP + FP} \tag{6.5}$$

$$revocação = \frac{TP}{TP + FN} \tag{6.6}$$

Finalmente, o F_1-*Score* é dado pela Equação 6.7.

$$F_1 = \frac{2 \times precisão \times revocação}{precisão + revocação} \tag{6.7}$$

6.2.2 Projeção e Aproximação de Variedade Uniforme

A Projeção e Aproximação de Variedade Uniforme (*Uniform Manifold Approximation and Projection*, UMAP) é uma técnica de redução de dimensionalidade[3] [79] que pode ser utilizada para visualização, de modo análogo a Incorporação Estocástica de Vizinhos com Distribuição t (*t-Distributed Stochastic Neighbor Embedding*, t-SNE) [74][4] e também permite combinar espaços com métricas diferentes [75]. Esta foi utilizada neste trabalho para visualizar a classificação de modelos quando aplicada a Perda Tripla.

6.2.3 Índice de Similaridade Estrutural

Um métrica particularmente utilizada para a seleção das quadros em vídeos foi o Índice de Similaridade Estrutural *(Structural Similarity Index, SSIM)* dada pela Equação 6.8 e que mede a diferença perceptual entre dois dados similares. Tendo sido desenvolvida para melhorar os métodos tradicionais, como a relação sinal-ruído de pico (*Peak Signal-to-Noise Ratio*, PSNR) e o Erro Quadrático Médio (*Mean Squared Error*, MSE) [13]. Essa métrica mostrou-se eficaz para comparação entre imagens com e sem *DeepFake* permitindo uma melhor seleção dos dados para compor as bases de dados.

$$\mathrm{SSIM}(X,Y) = \frac{(2\mu_X\mu_Y + C_1) + (2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)}, \tag{6.8}$$

onde μ_X é a média de $G(X)$, μ_Y é a média de Y, σ_X^2 é a variância de X, σ_Y^2 é a variância de Y, σ_{XY} é a covariância de $G(X)$ e Y, C_1 e C_2 são variáveis para estabilizar a divisão com o denominador. Neste trabalho, os valores de C_1 e C_2 foram computados como $(k_1 L)^2$ e $(k_2 L)^2$, respectivamente, onde L é a faixa dos valores de pixels (a diferença entre o maior e menor valor — normalmente $2^{\#bits\ por\ pixels} - 1$), $k1 = 0.01$ e $k2 = 0.03$, conforme proposto por Wang et al. [133].

6.3 Experimentos base

Nesta seção, detalhamos os experimentos realizados para as bases `FullTIMIT` (Subseção 6.3.1), `Sensitive` (Subseção 6.3.2), `Face Forensics plus plus` (Subseção 6.3.3), `Notre Dame Synthetic Face` (Subseção 6.3.4) e `DFDC` (Subseção 6.3.5). Utilizamos uma arquitetura inicialmente definida, em que escolhemos uma rede e a mantivemos totalmente descongelada. Em seguida, aplicamos uma camada *Global Average Polling* (GAP) [53] e finalizamos com uma camada densa [50] com dois nerônios e ativação *softmax*, visto que temos um problema binário, conforme representado na Figura 6.8.

Para os experimentos a seguir, utilizamos uma placa de vídeo gráfica *GPU GeForce GTX 1080 Ti* com 11GB de memória, além de uma *CPU Intel Xeon* com 72 núcleos e 500GB de memória RAM.

[3]Método de transformação de dados de um espaço em alta dimensão para uma representação de baixa dimensão, por meio da remoção de características redundantes presentes nos dados.

[4]Técnica para redução de dimensionalidade adequada para a visualização de conjuntos de dados de alta dimensão.

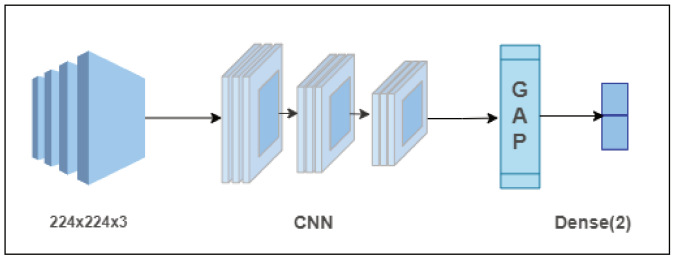


Figura 6.8: Representação da nossa arquitetura definida para os nossos experimentos base.

6.3.1 FullTimit

Para os experimentos realizados com essa base, iniciamos explorando diferentes classificadores tais como SVM e MLP. Então, com os resultados obtidos, decidimos empregar o aprendizado profundo. Para isto, utilizamos arquiteturas como a *ResNet50*[57] pré-treinada com o *ImageNet* para extração de características.

A fim de definir a arquitetura da rede final, aplicamos diferentes configurações de ajuste fino nas camadas finais, além de ajustes para selecionar o otimizador ideal para a configuração escolhida. Com isso, escolhemos os melhores resultados obtidos e os apresentamos na Tabela 6.1, conforme descrição de cada experimento. A referência completa de todos os experimentos realizados está no Apêndice A.1[5].

Experimento 1: Iniciamos nossos experimentos para detecção de *DeepFake*s, utilizamos o classificador SVM mantendo seus parâmetros com valores padrões.

Experimento 3: Como os resultados dos experimentos anteriores, utilizando classificadores simples, não geraram bons resultados, iniciamos a utilização de arquiteturas com aprendizado profundo. Escolhemos o *ResNet50* [6] aplicando sua taxa de aprendizado 0.1 padrão e com o otimizador *RMSprop*. Além disso, definimos o ajuste fino como sendo composto pelas camadas GAP e uma camada densa com dois neurônios e função de ativação *softmax* [51].

Experimento 6: Finalizamos mantendo as configurações do **Experimento 3**, tendo sido alterada a taxa de aprendizado para 0.001.

Experimento	F_1**-Score (%)**
1	51
3	51
6	100

Tabela 6.1: F_1*-Score* para os melhores experimentos realizados. Em destaque, a melhor configuração para esta base de dados.

Ao final dos experimentos com o `FullTIMIT`, foi possível definir que a melhor arquitetura para geração de novos modelos, é composta pela rede *ResNet50* com o otimizador *RMSprop*, taxa de aprendizado de 0.001 e os demais parâmetros com valores padrões.

[5]Optamos por manter a numeração dos experimentos para que o leitor possa comparar a evolução dos mesmos já que cada um faz uma modificação no anterior.

[6]As características especificas desta arquitetura podem ser conferidas em *resnet2020keras*

Quanto ao topo da rede, este é composto por uma camada *GlobalAveragePolling* seguida de uma camada densa com dois neurônios e ativação *softmax*. Logo, utilizaremos está configuração para iniciar os experimentos com a próxima base (`Sensitive`).

Com estes experimentos, identificamos que foi possível classificar as imagens entre real e sintéticas/falsas, quando empregamos redes neural profunda, devido a isto, optamos por não prosseguir com experimentos aplicando classificadores simples. Além disso, como o resultado final obtido atingiu a 100% de $F_1 - Score$, decidimos que não seria necessário continuar em busca de melhorias para a classificação desta base. Identificamos também que a quantidade de exemplos desta base de dados não oferece a representatividade necessária para a generalização de um modelo final, o que torna a tarefa de classificação relativamente simples. Logo, decidimos buscar novas bases e, como não havia nenhuma disponível naquele momento, criamos a nossa base para dar continuidade em nossos experimentos.

6.3.2 Sensitive

Iniciamos os experimentos com essa base de dados utilizando a arquitetura anteriormente definida, isto é, *ResNet50*. Nessa etapa, fixamos para todos os nossos experimentos um *batch* de tamanho 16 e utilizamos 10.000 imagens por classe, para manter o balanceamento das classes para treinamento. Descrevemos os melhores resultados obtidos a seguir e os apresentamos na Tabela 6.2.

Experimento 1: Com os pesos obtidos do melhor modelo treinado com o `FullTIMIT`, Subseção 6.3.1, testamos a predição das faces com o `Sensitive`.

Experimento 2: Em seguida, retomamos a configuração definida na Subseção 6.3.1, porém treinando com as imagens do `Sensitive`.

Experimento 3: Por fim, buscamos melhorar a classificação, alterando o otimizador, no qual utilizamos o *SGD* com taxa de aprendizado de 0.001.

Experimento	F_1**-Score (%)**
1	66
2	97
3	99

Tabela 6.2: F_1-*Score* para os melhores experimentos realizados. Em destaque, o melhor resultado obtido para o `Sensitive`.

Utilizar o modelo treinado com o `FullTIMIT` para predizer o `Sensitive` não apresentou bons resultados (66%) o que demostra que o `FullTIMIT` não possui a variabilidade necessária para generalização do modelo e confirma nossa suspeita de que a base `FullTIMIT` poderia ser considerada fácil. Quanto aos modelos treinados com `Sensitive`, o melhor método para detecção de *DeepFake*s foi obtido ao utilizar a mesma configuração do `FullTIMIT`, apenas alterando o otimizador.

Como os resultados obtidos indicam a correta classificação dos exemplos, decidimos não realizar novos experimentos com essa base. Além disso, por acreditarmos que essa base também não é suficientemente representativa e que pode provocar o enviesamento da

classificação, visto que, pelas faces sintéticas/falsas serem apenas de mulheres, exemplos de faces masculinas podem ser classificadas erroneamente como pertencente a classe real, dado que, durante o treinamento, o modelo não recebeu exemplos desse gênero. Por isso, continuamos a busca por uma base de dados mais representativa e que pudesse nos proporcionar a generalização dos modelos entre diferentes bases.

6.3.3 Face Forensics Plus Plus

Seguimos a mesma estrutura dos experimentos que as outras bases e utilizamos as melhores arquiteturas definidas anteriormente. Além disso, realizamos algumas explorações adicionais, como alterar o otimizador entre *SGD*, *RMSprop* e *Adam* [56], e varair as taxas de aprendizado. Como esta base contém divisões relacionadas as técnicas aplicadas para gerar os dados sintéticos/falsos, avaliamos o impacto que a técnica representa na classificação dos dados. Para isso, dividimos a classe sintética/falsa em: *DeepFake* *faceSwap* e ambas.

Apresentamos, na Tabela 6.3, os melhores resultados obtidos, enquanto a lista completa com todos os experimentos realizados estão descritas no Apêndice A.2.

Experimento 1: Com a configuração definida na Subseção 6.3.2, escolhemos treinar com ambas as técnicas de geração para *DeepFake*s.

Experimento 7: Decidimos alterar o otimizador, aplicando o *Adam* com uma taxa de aprendizado de 0.0001, tendo em vista que, conforme ocorreu com o `FullTIMIT`, alterar o otimizador foi capaz de proporcionar melhores resultados na classificação.

Experimento 8: Em seguida, mantivemos a estrutura do experimento anterior e treinamos utilizando apenas em imagens com *DeepFake*.

Experimento 9: Para fins comparativos realizamos novamente este experimento, porém considerando apenas em imagens com *faceSwap*.

Experimento	F_1**-Score (%)**
1	93
7	99
8	97
9	96

Tabela 6.3: F_1-*Score* para os melhores experimentos realizados. Em destaque, o melhor resultado obtido para esta base de dados.

Estes experimentos nos auxiliaram a confirmar que a melhor configuração da rede continua sendo a mesma anteriormente definida, sendo possível alcançar melhores resultados apenas empregando um otimizador diferente, neste caso o *Adam* com taxa de aprendizado de 0.0001.

Percebemos também que ao utilizar ambas técnicas para o treinamento, a classificação foi melhor do que ao aplicar apenas imagens com *DeepFake*s ou *faceSwap*, entretanto a diferença na classificação final não indica que a técnica seja capaz de influenciar negativamente os resultados, e, portanto, não é necessário desenvolver treinamentos específicos para cada tipo de técnica.

Ainda assim, continuamos em busca de bases mais desafiadoras, estando de acordo com o avanço das técnicas de produção de *DeepFake*s. Durante nossas buscas, encontramos uma base de dados que continha faces geradas sinteticamente e a utilizamos como meio para incrementar as classes sintéticas/falsa das outras bases. Detalharemos os experimentos utilizando esse incremento na seção seguinte.

6.3.4 Notre Dame Synthetic Face

Apesar desta base não ser especificamente para *DeepFake*s, a utilizamos por conter apenas faces geradas sinteticamente. Para isso, fizemos a junção desta base às classes sintéticas/falsas das bases anteriores, de modo a ampliar os exemplos dessa classe durante nossos treinamentos.

Assim como realizado nos experimentos anteriores, seguimos utilizando a configuração padrão da *Resnet50*, e mantivemos o último otimizador utilizado (*Adam*) com taxa de aprendizado de 0.0001. Dividimos os dados da classe sintéticas/falsas em que utilizamos 50% de imagens `NDSF` e os outros 50% da base de dados escolhida para o experimento, deste modo mantivemos o balanceamento entre as classes. Para melhor exemplificar, supondo que definíssemos que 20.000 faces reais seriam utilizadas, escolheríamos 10.000 da base `NDSF` e 10.000 de outra base de faces sintéticas/falsas. Ressaltamos que, para a etapa de validação, não utilizamos as imagens do `NDSF`. Apresentamos na Figura 6.9, os resultados obtidos após a junção do `NDSF` às outras bases.

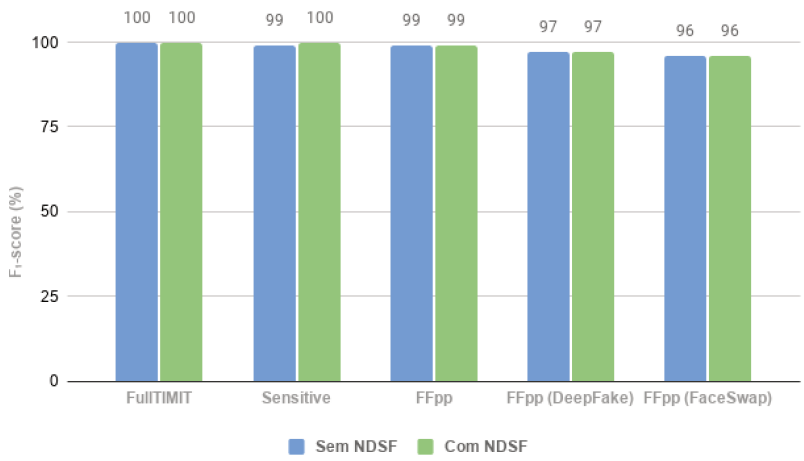


Figura 6.9: Comparamos o resultado obtido (em F_1-*Score*) com e sem a adição do `NDSF` ao conjunto de treinamento.

Observamos com estes resultados que, incrementar as bases de dados com o `NDSF` não proporcionou melhoria na classificação. Acreditamos que a possível razão para a não melhoria nas condições de treinamento é que a base `NDSF` tem um processo de criação sintético via Computação Gráfica, muito diferente dos processos de criação de *DeepFake*s atuais que se baseiam em aprendizado de máquina e redes generativas adversariais. Além disso, a melhoria apenas na classificação com o `Sensitive` pode indicar que a presença de faces masculinas, diminuiu o enviesamento do modelo, melhorando a representatividade para esta base.

Quanto ao experimento associado ao FullTIMIT, não podemos efetivamente analisar se houve alguma melhoria, tendo em vista, que já havíamos conseguido atingir uma classificação totalmente correta. Em relação ao FFpp, acreditamos que a base já contem variabilidade e, neste caso, adicionar as faces do NDSF não contribuiu para a classificação. Após analisarmos o comportamento do NDSF associado as demais bases, optamos por não continuar os experimentos com esta base e seguimos a utilização de uma nova base.

6.3.5 DeepFake Detection Challenge

Com a disponibilização do DFDC, iniciamos um protocolo de experimentos específicos para ele, de modo a atender as especificações do desafio. Isso se fez necessário pois, segundo as informações preliminares, haveriam algumas peculiaridades (como *DeepFake*s em áudio) que foram inseridas a fim de fornecer maior robustez ao DFDC, que intenciona tornar mais difícil a tarefa de classificação.

Além disso, por ser a maior base disponibilizada contendo vídeos com diferentes técnicas de *DeepFake* [47], decidimos abordar outras técnicas, que não foram aplicadas aos experimentos com as outras bases. Uma das modificações que realizamos, consistiu em utilizar entre 1 e 4 placas de vídeo gráfica *GPU Quadro RTX 8.000* com 48GB de memória, além de uma *CPU Intel Xeon Gold* com 80 núcleos e 1 TB de memória RAM. Isso nos possibilitou utilizar tamanhos de *batch* entre 500 e 4.500 imagens, tornando os treinamentos mais ágeis.

Decidimos iniciar nossos experimentos utilizando como arquitetura de rede a *MobileNet*, embasamos essa escolha na necessidade de produzir modelos compactos, tendo em vista que são necessários apenas 21 MB de espaço para armazenamento dos modelos produzidos com ela.

Partindo dessa escolha, realizamos experimentos para definir o melhor ajuste fino das camadas (Subseção 6.3.5). Em seguida, nos certificamos que utilizar a *MobileNet* seria suficiente para a classificação ao testarmos diferentes arquiteturas (Subseção 6.3.5). Como observamos pelos experimentos anteriores, a utilização do otimizador mais apropriado pode impactar significativamente na classificação, por isso, na Subseção 6.3.5, apresentamos experimentos com os principais otimizadores, associados a taxas de aprendizado que variam de 0.001 a 0.00001. Finalizamos nossos experimentos tradicionais analisando o impacto que o congelamento de camadas da rede, pode proporcionar ao resultado final da classificação (Subseção 6.3.5).

Camadas do Topo da Rede

Apesar de termos definido um ajuste padrão nos experimentos com as bases anteriores, vimos a necessidade de testarmos novas configurações, tendo em vista que o resultado obtido quando apenas replicamos as configurações anteriores, não apresentou um resultado satisfatório para esta base. Além disso, gostaríamos de analisar como a quantidade de camadas influencia o resultado final da classificação.

Diferentemente dos experimentos com as outras bases e como para o DFDC dispomos de uma quantidade maior de dados, decidimos utilizar 100 mil imagens por classe. Alteramos o otimizador para o *Nadam*, uma variação do *Adam*, em que o tempo necessário

para convergir é menor. Então, realizamos diversos experimentos, adicionando tanto camadas densa quanto *Dropout*, que estão integralmente descritos no Apêndice A.3.1. E, apresentamos na Tabela 6.4, as configurações com melhores resultados.

Experimento 1: Iniciamos nossas camadas mantendo a configuração utilizada nos experimentos com as outras bases, ou seja, foram adicionadas na saída da *MobileNet* uma *GAP*, seguida por uma camada densa com dois neurônios e ativação *softmax*.

Experimento 2: Em seguida, decidimos incluir uma camada densa com 256 neurônio e ativação *ReLU* antes da camada densa com dois neurônios e ativação *softmax*.

Experimento 4: Como obtivemos um resultado melhor, decidimos acrescentar mais uma camada. Desta vez, utilizamos uma camada densa com 512 neurônios antes da camada densa com 256 neurônios ambas com ativação *ReLU*, finalizando com a camada densa com dois neurônios e ativação *softmax*.

Experimento 9: Também avaliamos se desconsiderar a conexão entre alguns neurônios auxiliaria a classificação final. Para isso mantivemos a camada densa com 512 neurônios seguida da camada densa, com 256 neurônios e ativação *ReLU* e adicionamos um *Dropout*[7] [41] zerando as ativações de 50% dos neurônios antes de repassar os dados para a última camada densa, com dois neurônios e ativação *softmax*.

Experimento	Acurácia (%)
1	74.22
2	85.27
4	88.04
9	84.54

Tabela 6.4: Melhores configurações para o topo da rede. Em destaque, o melhor resultado obtido.

Ao final deste experimento, pudemos perceber que adicionar algumas camadas antes da classificação final, proporciona um melhor classificação dos dados. Por isso, definimos que seguiremos aos próximos experimentos adicionando uma camada *GAP*, seguida de uma camada densa com 512 neurônios, outra camada densa agora com 256 neurônios ambas com ativação *ReLU*, finalizando com a camada densa com dois neurônios e ativação *softmax*. Essa estrutura final está representada pela Figura 6.10.

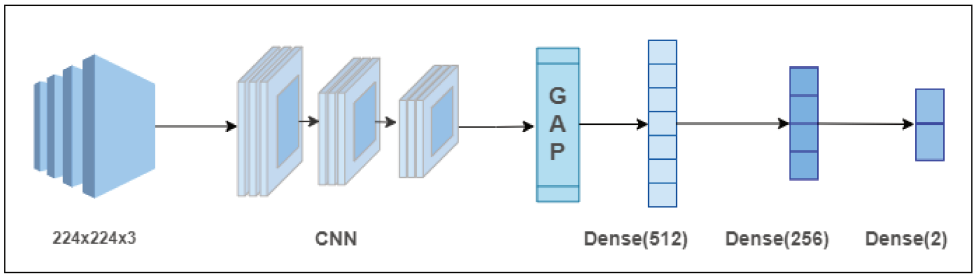


Figura 6.10: Configuração final com a melhor combinação de camadas para o DFDC.

[7] *Dropout* consiste em omitir aleatoriamente parte das ativações entre alguns neurônios entre camadas conectadas.

Seleção de Arquitetura

De acordo com nosso protocolo de experimentos, decidimos verificar se as arquiteturas de redes anteriormente escolhidas, realmente eram suficientes para a resolução do problema de classificação a que nos propomos resolver. Devido a isso, escolhemos algumas arquiteturas para realizar experimentos, comparando além da acurácia, se ocorre variações significante entre arquiteturas que possuem duas versões e, principalmente o tamanho do modelo produzido.

Todos estes experimentos foram treinados de acordo com a divisão dos dados entre versão 1 (considerando todos os vídeos) e versão 2 (com a seleção de vídeos pelo *SSIM*), conforme especificado na Subseção 6.1.4.

Utilizamos como entrada para a rede imagens de tamanho 224×224 e utilizando os pesos pré treinados da *ImageNet* com a rede totalmente descongelada e otimizador Nadam com os parâmetros padrão. Na Tabela 6.5 apresentamos as arquiteturas utilizadas e os resultados obtidos.

Arquitetura	**Versão 1 Acurácia (%)**	**Versão 2 Acurácia (%)**	**Tamanho do Modelo Gerado (MB)**
ResNet50 versão 1	80.87	81.38	108.9
ResNet50 versão 2	79.20	-	108.9
ResNet101 versão 1	82.34	83.89	185.4
ResNet101 versão 2	77.64	-	185.3
ResNet152 versão 1	84.81	77.19	248.6
ResNet152 versão 2	80.69	-	248.4
Xception	76.87	89.12	94
VGG16	73.33	71.31	61
VGG19	72.10	56.52	81
InceptionV3	71.69	83.33	101.9
InceptionResNetV2	73.84	89.66	219
DenseNet121	75.92	84.74	36.7
DenseNet169	77.31	85.27	63.4
DenseNet201	77.81	85.93	87.8
NASNet Mobile	65.61	50.00	26.7
MobileNet versão 1	80.37	86.65	21
MobileNet versão 2	75.64	-	18.8
*NASLarge**	58.47	57.87	368.1

Tabela 6.5: Relação de arquiteturas utilizadas. Em destaque, as arquiteturas que selecionamos para a próxima etapa de experimentos. *A rede *NASLarge* utiliza a entrada de 331×331.

Verificamos que o tamanho do modelo gerado, não sofre alteração significativa quando utilizamos diferentes versões da mesma arquitetura. Além disso, decidimos não realizar os experimentos com as segundas versões das arquiteturas, para os experimentos com a versão 2 da divisão dos dados, pois, conforme observamos, não houve melhora significativa entre as versões das arquiteturas para sustentar a continuidade de experimentos com elas.

Observamos ainda que, para o nosso problema, as arquiteturas com melhor acurácia foram: *ResNet152* (melhor entre a família *Resnet*) e a *MobileNet*, considerando a pri-

meira versão da divisão dos dados, porém, decidimos basear as nossas decisões apenas pelo resultados obtidos na segunda versão, visto que minimizamos as classificações errôneas ao desconsiderar vídeos rotulados como falsos/sintéticos mas que não apresentavam diferenças estruturais, sugestivos para casos com alteração apenas no áudio.

Analisando os resultados para a segunda versão da divisão dos dados, identificamos que as melhores arquiteturas foram a *InceptionResNetV2*, *Xception* e a *MobileNet*. Entretanto, apesar do resultado obtido com a *InceptionResNetV2* ter sido levemente superior, decidimos considerar apenas a *Xception* devido, principalmente, ao tamanho do modelo gerado. E, apesar de a *MobileNet* não ter apresentado os melhores resultados, decidimos utilizá-la devido a sua arquitetura de camadas que minimiza consideravelmente o tempo necessário para treinamentos, além de ocupar menos espaço de armazenamento.

Após escolhermos as arquiteturas, buscamos encontrar a melhor combinação de otimizador e taxa de aprendizado que, como observado nos experimentos com as outras bases, identificar a melhor relação arquitetura de rede — otimizador — taxa de aprendizado, pode variar significativamente o desempenho na classificação.

Seleção de Otimizador e Taxa de Aprendizado

Visando analisar o comportamento das redes em relação às diferentes taxas de aprendizado e otimizadores, realizamos os próximos experimentos considerando a *MobileNet* e variando entre os otimizadores *Nadam*, *Adam*, *Adamax*, *RMSProp* e *SGD*. Além de taxas de aprendizado que vão de 0.001 a 0.00001. E, apresentamos na Tabela 6.6 a relação entre otimizador — taxa de aprendizado e os resultados obtidos.

Otimizador	**Taxa de Aprendizado**	**Acurácia (%)**
Adam	0.001	87.89
	0.0001	87.79
	0.00001	85.32
Adamax	0.001	81.42
	0.0001	87.14
	0.00001	87.01
Adagrad	0.001	79.41
	0.0001	86.95
	0.00001	82.39
Nadam	0.001	86.21
	0.0001	85.54
	0.00001	87.24
RMSProp	0.001	73.42
	0.0001	86.84
	0.00001	79.51
SGD	0.001	79.42
	0.0001	60.86
	0.00001	83.42

Tabela 6.6: Experimentos realizados para definir a relação entre otimizador e taxa de aprendizado. Em destaque, os melhores resultados obtidos.

Observamos que utilizar *Adam* ou suas variações proporcionam os melhores resultado e que, apesar de terem o mesmo principio, a melhor taxa de aprendizado varia, sendo necessário ser identificada e ajustada, de acordo com as especificidades do problema a ser resolvido.

Complementarmente, reforçamos a análise explorando as arquiteturas *ResNet152* e *Xception* com os otimizadores *Adam*, *Adamax* e *Nadam* com as taxas de aprendizado em que apresentaram melhores resultados, e descrevemos na Tabela 6.7 a acurácia obtida para cada um desses experimento.

Arquitetura	**Otimizador**	**Taxa de Aprendizado**	**Acurácia (%)**
ResNet152	*Adam*	0.0001	80.20
	Adamax	0.001	82.97
	Nadam	0.001	76.88
Xception	*Adam*	0.0001	88.70
	Adamax	0.001	87.41
	Nadam	0.001	89.18

Tabela 6.7: Melhores resultados de otimizador e taxa de aprendizado, aplicados à *ResNet152* e *Xception*. Em destaque, o melhor resultado obtido para cada arquitetura.

Após estes experimentos, concluímos que seguiremos utilizando o *Nadam*, com taxa de aprendizado de 0.00001, visto que com ele obtivemos os melhores resultado em duas arquiteturas distintas. Com isso, seguiremos ao último conjunto de experimentos tradicionais, em que analisaremos o impacto do congelamento das camadas da rede, para a classificação.

Taxa de Congelamento

A taxa de congelamento é importante para definir quantas e quais camadas da CNN pré-treinada terão seus pesos atualizados durante o treinamento do modelo. Para os próximos experimentos, mantivemos o otimizador *Nadam* com taxa de aprendizado de 0.00001 ajustados a *MobileNet* e variamos a taxa de congelamento da rede entre totalmente descongelada (1.00) à apenas o topo descongelado. Apresentamos na Tabela 6.8 as taxas escolhidas e os resultados obtidos.

Congelamento	**Acurácia (%)**
Topo	86.51
0.25	85.76
0.50	85.63
0.75	83.32
0.90	85.47
1.00	88.04

Tabela 6.8: Experimentos variando a taxa de congelamento da rede, ou seja, quanto da rede será utilizado para o o ajuste do topo da rede. Em destaque, o melhor resultado obtido.

Estes resultados sugerem que ao utilizar apenas as camadas finais da rede, obtemos boa parte das principais características utilizadas para a classificação. Entretanto, como o melhor resultado foi encontrado ao manter todas as camadas da rede, acreditamos que algumas características importantes são extraídas nas camadas iniciais. Logo, decidimos seguir utilizando todas as camadas da rede para o treinamento.

Ainda para fins de comparativos, treinamos a *ResNet152* e o *Xception* mantendo todas as camadas descongeladas e obtivemos 87.56% e 89.37%, de acurácia, respectivamente. Após finalizarmos estes testes, definimos todas as configurações referentes à arquitetura do modelo para treinamento e, como acreditamos poder melhorar a classificação, partimos para a utilização de aumentação dos dados durante o treinamento, conforme seção a seguir.

6.4 Método 1: Aumentação dos Dados

Aplicar aumentação nos dados é uma técnica eficaz e comumente utilizada tanto para aumentar a quantidade de exemplos, quanto para adicionar variabilidade aos dados.

Em nosso trabalho, utilizamos a aumentação dos dados com o intuito de apresentar para a rede exemplos em diferentes condições. Para tanto, adicionamos aumentações clássicas tanto geométricas quanto radiométricas, sendo as mais utilizadas detalhadas a seguir. Mantivemos a arquitetura definida na Subseção 6.3.5, para realizarmos estes experimentos e adicionamos as aumentações apenas aos exemplos de treinamento.

Rotação: rotacionamos as imagens de face em até 90 graus.

Ampliação (*zoom*): ampliamos a imagem em um intervalo entre 20 e 80%.

Espelhamento (*flip*): a imagem foi invertida horizontal ou verticalmente.

Brilho: adicionamos brilho a imagem, variando entre 20 e 100%.

Iniciamos nossos experimentos aplicando as aumentações de rotação, translação (*shift*) na largura e na altura de até 20%, ampliação, espelhamento horizontal e vertical, brilho, cisalhamento de até 20% e mudança de canal em 10 posições. Nomeamos esse experimento de *7 Augs*, em referência a quantidade de técnicas aplicadas.

Como esperado, aplicar técnicas que provocassem deformações na imagem não contribuiu para a classificação, pois parte dos artefatos introduzidos pela técnica de produção de *DeepFake*s podem ter sido modificadas ou perdidas. Por isso, diminuímos as aumentações aplicando rotação, ampliação, espelhamento vertical, brilho e cisalhamento de até 20% e nomeamos esse experimento como *5 Augs*. Apresentamos os principais resultados obtidos ao utilizarmos essa abordagem, na Tabela 6.9.

Conforme realizamos nossos experimentos, observamos que aplicar os diferentes tipos de aumentação não auxiliam na correta classificação dos nossos exemplos. Isso se deve ao fato de que buscamos identificar artefatos nas imagens e, ao aplicarmos aumentação, acrescentamos novas informações. Nos casos em que adicionamos espelhamento ou brilho, os resultados foram melhores, devido ao tipo de informação que aplicar cada uma dessas técnicas acrescenta a imagem.

Ao compararmos a classificação com e sem aumentação, identificamos que incrementar

Técnica	**Acurácia (%)**
7 Augs	70.87
5 Augs	80.23
Espelhamento + Brilho	85.28
Rotação + Ampliação	81.38
Espelhamento	84.38
Brilho	85.19

Tabela 6.9: Experimentos realizados para encontrar a melhor configuração dos dados. Em destaque, o melhor resultado obtido.

nossos dados não apresentou melhoria em relação ao nosso resultado anterior, 85.28% e 88.04% de acurácia, respectivamente. Com isso, optamos por não utilizar aumentação nos dados e decidimos aplicar técnicas de transformação dos dados de entrada, como *Patches* (Subseção 6.5.1) e Mapa de Profundidade (Subseção 6.5.2).

6.5 Método 2: Representações de Entrada

Aplicamos aos experimentos desta seção modificações da imagem da face original. Com isso, buscamos analisar o comportamento da rede, ao lhe apresentarmos dados não convencionais, como *Patches* específicos da face (Subseção 6.5.1) ou mapas de representação da profundidade — em que as informações de distância entre os componentes da imagem — (Subseção 6.5.2). Além disso, realizamos experimentos associando essas novas representações e a face original.

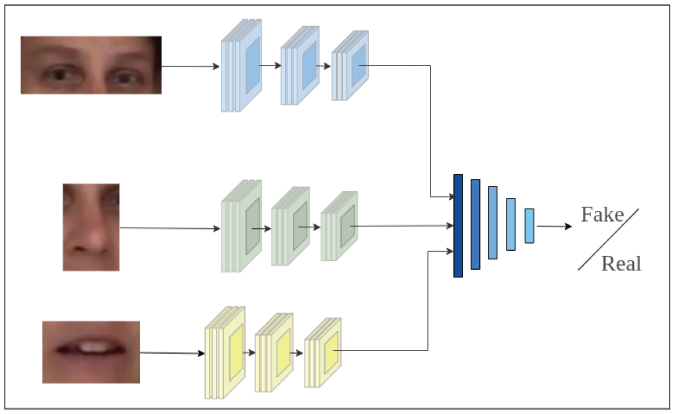
6.5.1 Patches

Para a abordagem utilizando os *patches*, foi necessário realizar uma etapa de extração das regiões específicas, conforme descrito na Subseção 6.1.3. Decidimos extrair apenas os olhos, o nariz e a boca, visto que as demais informações da face podem ser adquiridas ao utilizar a imagem original. Adicionalmente, algumas técnicas para a produção de *DeepFake*s introduzem artefatos nestas regiões.

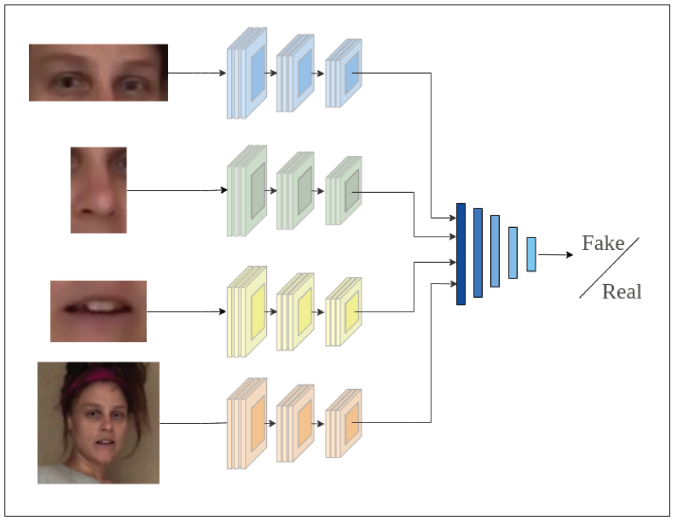
Em seguida, dividimos nossos experimentos, em concordância com os tipos de imagens utilizadas, ou seja apresentando para a rede apenas os *patches*, ou associando *patches* e face.

Mantivemos a configuração anteriormente definida na Subseção 6.3.5. Entretanto, houve a necessidade de realizamos uma alteração na estrutura da rede, tendo em vista que agora é necessário adotar uma rede para cada tipo de entrada. Descrevemos essa nova estrutura conforme o esquema da Figura 6.11 (a), quando utilizado apenas os *patches* e na Figura 6.11 (b), quando associamos os tipos de dados.

Para o desenvolvimento dos nossos experimentos, dividimos em duas abordagens. Na primeira, utilizamos apenas os *patches* como entrada para a rede e descrevemos na Tabela 6.10, os resultados para cada tipo de *patch*. Enquanto que para os experimentos incluindo também a face (nossa segunda abordagem), os descrevemos na Tabela 6.11.



(a)



(b)

Figura 6.11: Representação da rede para o treinamento utilizando *patches*. Na abordagem (a), são considerados apenas os *patches*, enquanto em (b), é adicionada a face referente aquelas regiões. Então, em ambas arquiteturas, realizamos a concatenação das características extraídas e as ajustamos para a classificação.

Região	**Média**
Olhos	76.66
Nariz	72.31
Boca	68.50
Olho e Nariz	73.42
Olho e Boca	70.43
Nariz e Boca	71.04
Olho, Nariz e Boca	83.66

Tabela 6.10: Experimentos utilizando apenas os *patches* - nossa primeira abordagem. Para estes experimentos, consideramos os *patches* individuais e a combinação entre eles.

Podemos observar que em ambas abordagens o melhor resultado obtido foi ao utilizarmos todos os *patches* combinados. Além disso, utilizá-los em conjunto com a face original apresentou melhores resultados se comparado a quando aplicamos apenas *patches*. Entretanto, adotar essa abordagem não trouxe melhoria ao nosso modelo (Subseção 6.3.5), com

Região	Acurácia
Olhos e Face	84.91
Nariz e Face	86.49
Boca e Face	85.24
Olho, Nariz e Face	86.77
Olho, Boca e Face	85.52
Nariz, Boca e Face	85.79
Olho, Nariz, Boca e Face	86.94

Tabela 6.11: Experimentos utilizando a combinação de *patches* e faces - nossa segunda abordagem.

o qual havíamos obtido 88.04% de acurácia. Com isso, decidimos realizar experimentos com outra representação de entrada.

6.5.2 Mapa de Profundidade

Assim como realizado para a representação anterior, o nosso primeiro passo consistiu em gerarmos os mapas de profundidade para as faces, um exemplo de mapa é apresentado na Figura 6.12.

(a) (b)

Figura 6.12: Imagens geradas a partir do Mapa de Profundidade. (a) Imagem real e (b) Imagem sintética/falsa.

Com a configuração de rede definida na Subseção 6.3.5, aplicamos uma alteração na sua entrada, de modo a receber o mapa de profundidade. Devido ao mapa estar em escala de cinza, o replicamos a fim de preencher os 3 canais esperados pela rede. Após esta adaptação realizamos o treinamento e obtivemos 53.70% de acurácia.

Em seguida, similar ao que realizamos nos experimentos com *patches* decidimos adicionar um canal extra para entrada da rede de modo que a CNN receberá como entrada 4 canais, RGB + mapa, conforme esquema apresentado na Figura 6.13.

Como resultado deste experimento obtivemos 85.24% de acurácia. E, conforme observado, quando a imagem original não é utilizada no treinamento, a classificação obtém resultados inferiores. Constatamos que é necessário à rede ser exposta a representação facial original durante o treinamento. Além disso, associar a face original com outras representações, também não representou melhoria na classificação, em relação ao nosso modelo base.

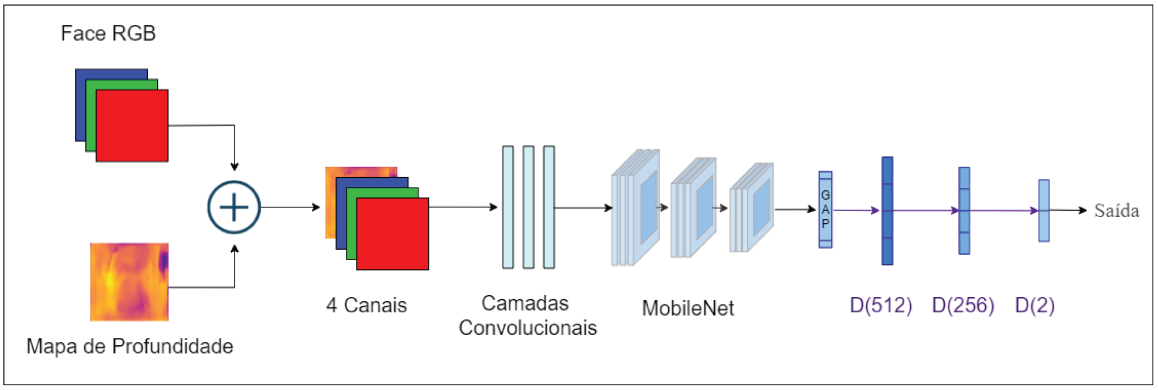


Figura 6.13: Esquema final para a rede modificada para receber o mapa de profundidade e a face.

Buscamos outra alternativa para avaliar nossos dados, com isso, decidimos realizar nossos próximos experimento associando a nossa função de custo tradicional de entropia cruzada à perda tripla.

6.6 Método 3: Perda Tripla

Neste método, combinamos a função de custo de entropia cruzada, com a perda tripla. Acreditamos que associar esta função, poderia melhorar nossa classificação, dada a sua propriedade de maximizar a distância de exemplos de classes diferentes enquanto aproxima exemplos da mesma classe.

Para os nossos treinamentos, dividimos a rede em duas partes. A primeira parte (rede A) consiste em uma *MobileNet* pré-treinada, sem as camadas do topo e com a função de perda tripla semi-difícil sendo aplicada diretamente nos vetores de características. A segunda parte (rede B — considerada para a rede classificadora) corresponde às camadas do topo da rede dos experimentos anteriores.

O treinamento ocorreu em duas etapas: inicialmente, treinamos a rede A e computamos a função de perda tripla e, em seguida, congelamos todas as suas camadas e fizemos o treinamento da rede B. Descrevemos a seguir os experimentos que obtiveram melhores resultados, sendo seus valores apresentados na Tabela 6.12. A referência completa de todos os experimentos, está no Apêndice A.3.2.

Experimento 2: Iniciamos nossos experimentos definindo que 40% da rede A serão mantidos descongelada. Em seguida, alteramos as camadas dessa rede, acrescentando uma camada densa com 1024 neurônios e ativação $ReLU$ antes de uma camada densa com 512 neurônios sem ativação e incluímos uma camada de normalização L2. Enquanto para a rede B, uma camada densa com 512 neurônios e uma camada densa 256 ambas com ativação $ReLU$ e uma camada densa com um neurônio e ativação *sigmoid*. Para cada 5 épocas da rede A, treinamos 1 época da rede B.

Experimento 8: Analisando a *loss* da rede A, acreditamos ser possível obter um valor ainda inferior e que, quando combinado com o resultado da rede B, possa haver um ponto de equilíbrio. Por isso, zeramos as ativações de 50% dos neurônios após a camada densa com 1024 neurônios e ativação $ReLU$ e mantivemos a camada densa de 512 neurônios

sem ativação, seguida de uma camada de normalização L2. Já para a rede B mantivemos apenas uma camada densa de um neurônio e ativação *sigmoid*.

Experimento 11: A fim de analisar o impacto que os pesos compartilhados, alteramos a taxa de congelamento da rede A, mantendo-a totalmente descongelada.

Experimento	**Rede A loss**	**Rede B acc (%)**
2	0.9195	80.26
8	0.8947	80.88
11	0.9027	88.09

Tabela 6.12: Resumo dos experimentos realizados com a perda tripla. Em destaque a o resultado da melhor acurácia.

Com esses resultados, acreditamos que o aumento em 0.008 na *loss* da rede A, seja aceitável, tendo em vista o aumento em cerca de 7.21 pontos percentuais na classificação final (considerando os experimentos 8 e 11).

Entretanto, este resultado não representou melhora significativa no nosso melhor modelo para o `DFDC`, sendo 88.09% de acurácia para esta abordagem e 88.04% definido na Subseção 6.3.5. Logo, resolvemos reanalisar os dados do `DFDC`, pois, vimos que ao aplicar diferentes técnicas a classificação não se tornou mais confiável e acreditamos que alguns dados podem estar influenciando a correta classificação, já que não temos uma rotulação que nos permita desconsiderar *DeepFake*s em áudio.

6.7 Reanálise da Base de Dados `DFDC`

Analisando os resultados dos experimentos realizados com o `DFDC`, decidimos reanalisar os vídeos selecionados para compor à base final que utilizamos nos experimentos. Isso nos resultou na versão 3 da separação dos dados entre treino e validação, conforme Subseção 6.1.4.

Então, utilizamos as configurações do nosso melhor modelo para esta base, definido na Subseção 6.3.5, e treinamos um novo modelo considerando os novos dados da versão 3. Com esta abordagem, obtivemos a acurácia de 98.10%, sendo este o melhor resultado para o `DFDC`. E, a fim de comparar os resultados obtidos para cada base de dados, além de disponibilizar um resumo com as técnicas aplicadas ao `DFDC`, desenvolvemos a seção a seguir.

6.8 Comparação entre os Métodos e Análise dos Resultados

Iniciamos nossa análise comparativa considerando os diferentes métodos que aplicamos a base `DFDC` e apresentaremos na Figura 6.14 cada um deles. Como podemos observar, aplicar a nossa técnica base, nos proporcionou resultados superiores aos demais. Acreditamos que isso se deve, a característica do problema em que as informações relevantes

para a classificação podem ser obtidas a partir da imagem original, sem a necessidade de complementação. Ressaltamos que nestes experimentos o limiar do SSIM em que utilizamos, pode ter colaborado para que a tarefa se tornasse mais complexa, uma vez que dados rotulados como sintéticos/falsos e que não fossem efetivamente *DeepFake*s, podem ter sido selecionados.

Como método proposto para minimizar a incorreta seleção de vídeos sintéticos/falsos sem *DeepFake*s, reanalisamos toda a base, coletando informações estatísticas mais precisas, além de selecionar empiricamente um novo limiar para o SSIM, com base em amostras de possíveis limites superiores, ou seja, visualmente verificamos o par de vídeos real-falso para determinar quando a diferença visual se tornava insignificativa. A partir disto, foi possível identificar que na maioria dos casos em que a diferença identificada pela comparação por SSIM, ocorreram em cenários com pouca iluminação, pela baixa qualidade do vídeo ou por se tratarem de pessoas com tons de peles escuros. Utilizaremos a Figura 6.15 para exemplificar visualmente como houve variabilidade na classificação entre nossas 3 versões de divisão dos dados para o `DFDC`.

Por fim, analisamos qual o resultado para a tarefa de classificação considerando cada base em que utilizamos. Por meio da Figura 6.16, podemos observar que os modelos finais gerados conseguem classificar satisfatoriamente os exemplos desenvolvidos com técnicas conhecidas durante o treinamento. Entretanto, parte do nosso problema consiste em identificar se é possível alcançar a generalização do modelo, de modo que este seja capaz de classificar corretamente quando exposto a técnicas desconhecidas.

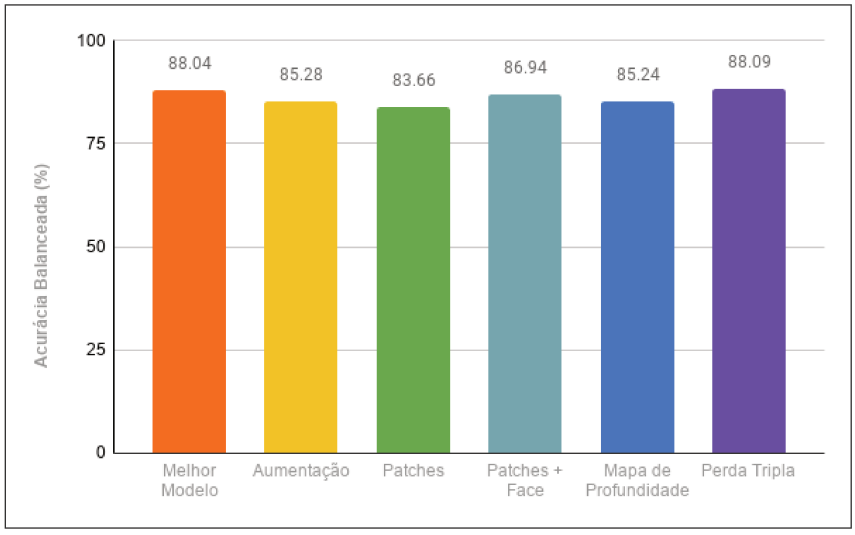


Figura 6.14: Apresentamos uma comparação entre os diferentes métodos utilizados para a classificação da base `DFDC`, considerando a sua segunda versão de divisão dos dados.

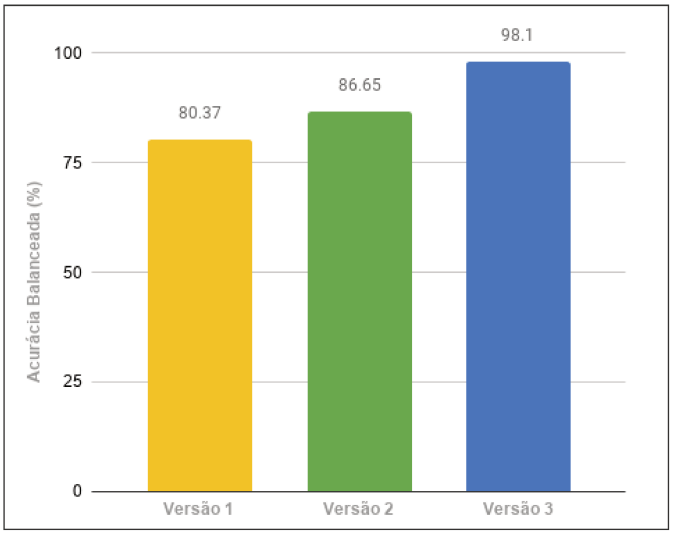


Figura 6.15: Variabilidade nos resultados da classificação entre as três versões, aplicadas ao método base para o DFDC

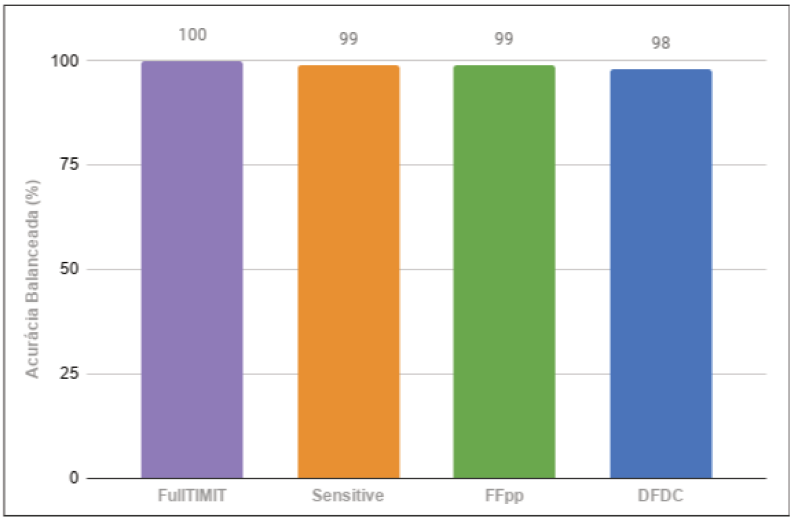


Figura 6.16: Comparativo entre o resultado da classificação com cada base de dados, considerando a acurácia balanceada.

6.9 Teste Cego e Comparação com a Literatura

Finalizamos os nossos experimentos, verificando a robustez dos modelos gerados em que serão utilizados para predizer os dados desconhecidos da base `DFD - Google`. Além disso, compararemos os resultados obtidos pelos nossos melhores experimentos para cada base, com os desenvolvidos pela literatura.

6.9.1 Teste Cego com o DFD - Google

Visando avaliar a performance dos nossos métodos para classificação, selecionamos um subconjunto da base de dados `DFD - Google`, que não foi utilizada em nenhum dos experimentos anteriores, para ser predita pelos nossos melhores modelos de cada base de dados (Figura 6.17). Ao todo, utilizamos 30.669 faces desta base, sendo 5.564 reais e 25.105 sintéticas/falsas.

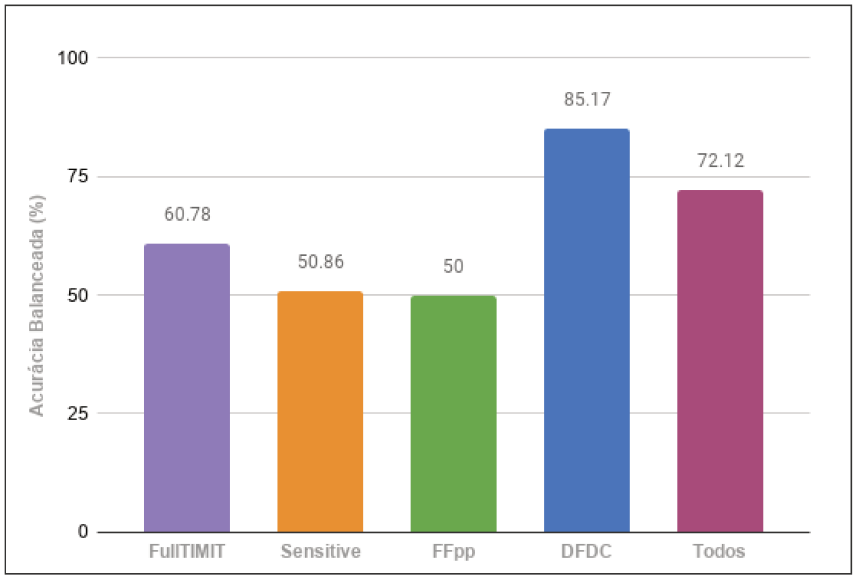


Figura 6.17: Resultado obtido a partir da predição do `DFD - Google` com melhor modelo gerado para cada base de dados.

Ao utilizarmos o melhor modelo treinado com o `FullTIMIT` para a predição obtivemos 60.78% de acurácia balanceada. Conforme apresentada na matriz de confusão (Figura 6.18) cerca de 62% dos exemplos sintéticos/falsos foram classificados erroneamente como reais. Isso indica que o modelo desenvolvido como esta base, não pode ser generalizado para classificar os exemplos sintéticos/falsos de outras bases. Assim como previámos, visto que o `FullTIMIT` contém exemplos em cenários controlados e com pouca variabilidade de posições da face.

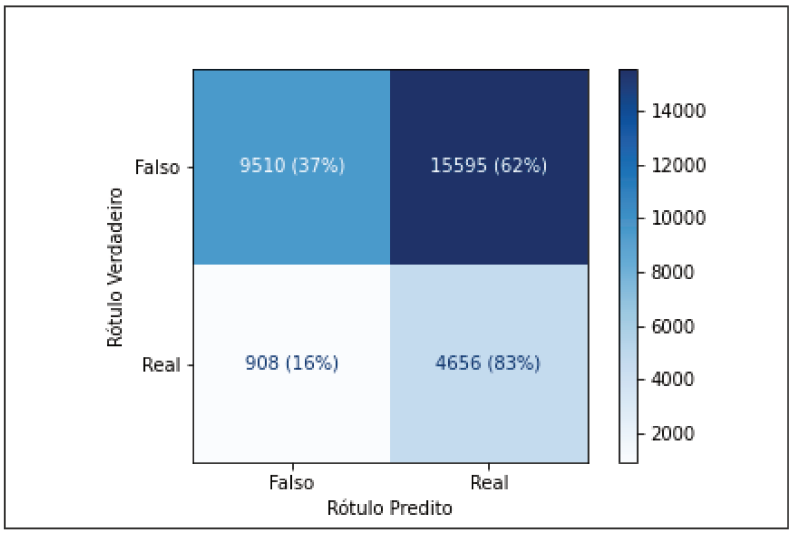


Figura 6.18: Matriz de confusão gerada a partir da predição do `DFD - Google` com o melhor modelo da base de dados `FullTIMIT`.

Para a predição utilizando o modelo do `Sensitive` a acurácia balanceada foi de apenas 50.86% e, conforme a matriz de confusão da Figura 6.19, percebemos que 95% das imagens sintéticas/falsas foram erroneamente classificadas, o que indica a necessidade de maior variabilidade de exemplos para tornar este modelo robusto. Esse problema também era

esperado tendo em vista que uma das maiores limitações desta base consiste na falta de representatividade do gênero masculino.

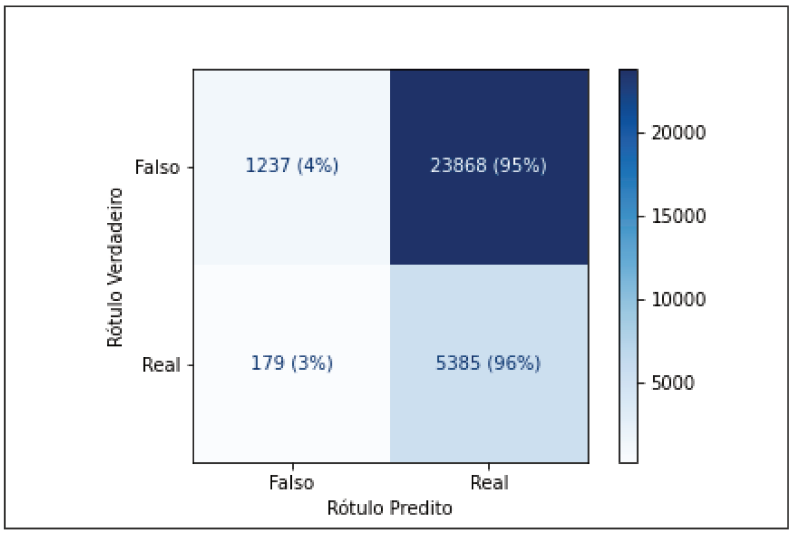


Figura 6.19: Matriz de confusão gerada a partir da predição do `DFD - Google` com o melhor modelo da base de dados `Sensitive`.

Novamente obtivemos 50% de acurácia balanceada. Entretanto, para esta predição utilizamos o modelo gerado a partir do `FFpp`. E, analisando a matriz de confusão da Figura 6.20, percebemos que apesar de 99% dos exemplos sintéticos/falsos terem sido corretamente classificados, o modelo classificou corretamente apenas 2 exemplos como reais. Com isso, acreditamos ser necessário analisar quais características presentes nas imagens reais podem proporcionar tal efeito. Uma das possibilidades é que, como esta base utiliza diferentes taxas de compressão, a rede tenha aprendido erroneamente que imagens de baixa resolução se referem a imagens manipuladas e, portanto, as classifique como sendo falsas.

Selecionamos o nosso melhor modelo para o `DFDC` (apresentado na Subseção 6.7) que obteve aproximadamente 98% de acurácia e realizamos a predição no teste cego para o `DFD - Google`. Com esta combinação, o resultado atingiu 85.17% de acurácia balanceada. E, com a matriz de confusão apresentada pela Figura 6.21, nos foi possível analisar que apenas 19% dos exemplos sintéticos/falsos foram erroneamente classificados como reais, o que demonstra maior robustez na classificação para uma base nunca antes utilizada, em comparação com as demais predições. Isso nos permite constar que o nosso modelo foi capaz de generalizar, mas ainda deixa espaço para melhoria dado que temos cerca de 17 erros de classificação para cada 100 casos.

Acreditamos ser possível obter um resultado ainda melhor ao combinar as bases de dados `FullTIMIT`, `Sensitive`, `FFpp` e `DFDC` para produzir um novo modelo. Devido a isto, decidimos produzir um último modelo, em que combinamos todas as bases. Para este, tivemos como resultado 72.12% de acurácia balanceada ao utilizarmos para predizer a base `DFD - Google`. Esse resultado inferior ao obtido anteriormente, quando utilizamos apenas o modelo com `DFDC` e, conforme analisado na Figura 6.22, apesar de obtermos 88% de acerto para a classe sintéticos/falsos, diminuímos a eficiência da classificação de imagens reais (acertando apenas 44%, aproximadamente).

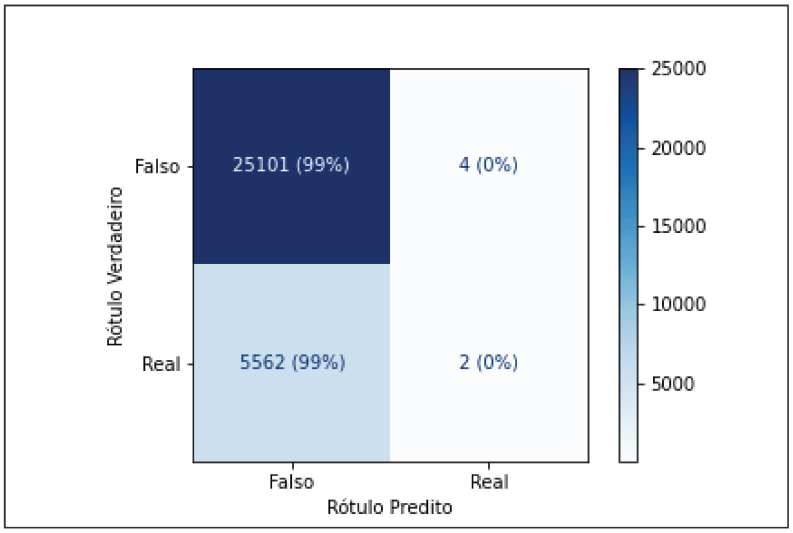


Figura 6.20: Matriz de confusão gerada a partir da predição do `DFD - Google` com o melhor modelo da base de dados `FFpp`.

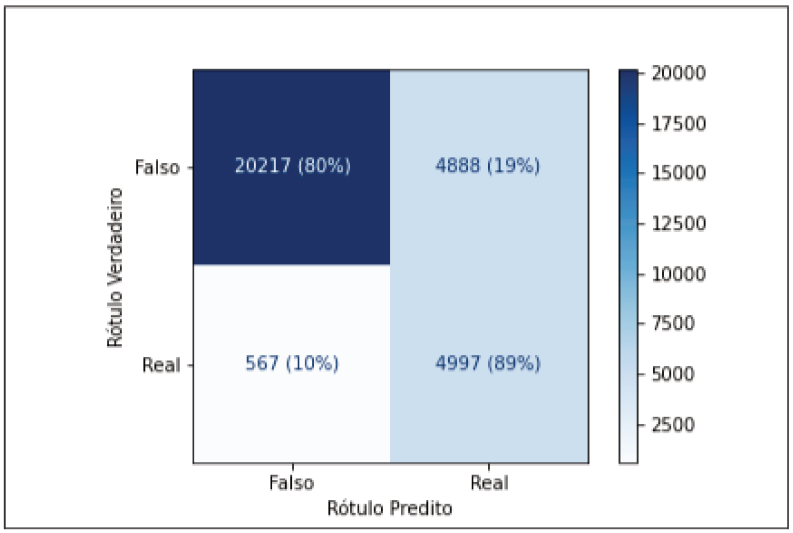


Figura 6.21: Matriz de confusão gerada a partir da predição do `DFD - Google` com o melhor modelo da base de dados `DFDC`.

6.9.2 Comparação com a Literatura

Nesta subseção, nós comparamos o resultado obtido pelo nosso método, em que desenvolvemos um modelo treinado apenas com a base `DFDC` e alguns trabalhos da literatura. Para isso, avaliaremos a robustez de cada método a partir do teste cego com o `DFD - Google`.

Para esta avaliação, selecionamos quatro trabalhos Li e Lyu [67], Nguyen et al. [83], Rössler et al. [99] e os apresentamos a partir da técnica aplicada sendo: *FWA*, *DSP-FWA*, Cápsula e *Xception*, conforme apresentado no Capítulo 3. Decidimos utilizá-los pois outros autores também os utilizaram para embasar a qualidade dos resultados obtidos.

Para estes experimentos, identificamos o repositório de cada técnica e seguimos as instruções dos autores para correta utilização dos modelos pré treinados. Com esses modelos,

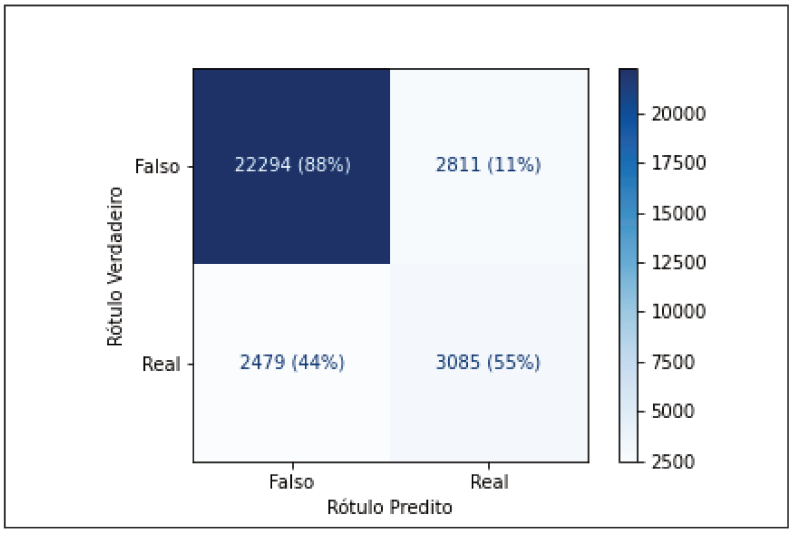


Figura 6.22: Matriz de confusão gerada a partir da predição do `DFD - Google` com o modelo gerado a partir da junção de todas as bases de dados.

realizamos a predição da base de dados escolhida, que também não foi utilizada durante o treinamento de nenhum dos métodos, garantindo assim a propriedade de avaliação entre diferentes bases de dados. Ressaltamos que o método cápsula, foi pré-treinado na base `FFpp`, incluindo seus quatro tipos de dados sintéticos/falsos: *DeepFake*s, *Face2Face*, *FaceSwap* e Texturas Neurais. Apresentamos os resultados obtidos por meio da Figura 6.23.

Podemos perceber que com o método *FWA*, em que são detectados distorções deixadas pelo processo de geração de *DeepFake*s obtivemos 63.90% de *AUC* (métrica que escolhemos utilizar, visto que a literatura comumente a utiliza para análise dos resultados). Apresentando uma melhora significativa ao método, quando adicionadas características para simular diferentes resoluções (DSP-FWA). Ao utilizar a abordagem em Cápsula, foi obtida uma classificação similar ao *FWA*, o que nos indica que métodos mais complexos não agregam melhoria na robustez do modelo. Por fim, treinar um modelo apenas apresentando os dados a uma *Xception* apesar de eficaz, ainda permite que melhorias sejam exploradas. Similar a isto, ao associarmos uma rede de aprendizado profundo com ajuste de algumas camadas e a dados mais representativos, como o nosso método proposto, é possível obter um método mais robusto e generalizável.

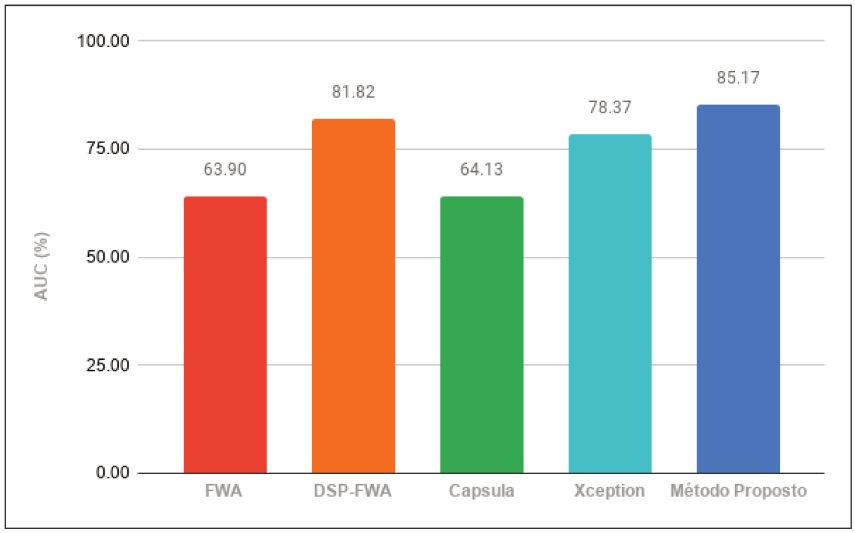


Figura 6.23: Comparação de Resultados obtidos aplicando o teste cego, por meio da predição da base `DFD - Google` com os modelos pré-treinados de métodos propostos pela Literatura e o nosso método proposto.

6.10 Considerações Finais

Ao final de nossos experimentos, identificamos que os métodos que decidimos utilizar foram razoáveis para a correta classificação das imagens. No entanto, ainda é preciso pesquisa na área, uma vez que ainda existe cerca de 15% de erro na classificação entre bases de dados.

Analisando os nossos experimentos base, em que aplicamos à rede de aprendizagem profunda dados gerados por diferentes técnicas, pudemos observar que é possível desenvolver um modelo capaz de realizar a correta classificação de quase 100% das imagens. Apenas quando utilizamos o *DFDC*, necessitamos buscar métodos alternativos para auxiliar na tarefa de classificação. Acreditamos que parte da dificuldade que encontramos para a detecção de imagens com *DeepFake*, se deve a característica da base de também considerar *DeepFake*s de áudio, sendo necessário o refinamento da rotulação para desconsiderar estes casos. Assim que ajustamos os dados para minimizar a ocorrência de *DeepFake*s não faciais, conseguimos atingir uma melhor classificação dos dados.

Reiteramos que aplicar diferentes técnicas para incrementar as imagens do *DFDC*, não proporcionaram melhoria para a classificação. Isso se deve à características do problema e da solução proposta, em que buscamos encontrar inconsistências ou artefatos inseridos nas imagens *DeepFake*s e, quando adicionada novas informações podemos inserir artefatos irrelevantes ou mascarar as propriedades originais da imagem.

Na nossa última abordagem, ajustar o modelo para separar as classes no hiper-espaço e então classificá-las, apresentou uma melhoria miníma na classificação final dos exemplos, em comparação com a abordagem base, ao passo que aumenta a complexidade e tempo do treinamento.

Finalizamos nossos experimentos avaliando a robustez dos modelos, quando inseridos em cenários desconhecidos. Para isto, realizamos a predição do `DFD - Google` tanto para os modelos gerados por nós quanto por trabalhos da literatura. Com isso, pudemos

observar que os modelos gerados a partir de bases com maior variabilidade, desempenham melhor a tarefa de classificação, sendo mais robustos em cenários desconhecidos.

Capítulo 7

Conclusões

Neste capítulo, reavaliamos nossas descobertas e os principais tópicos discutidos nesta dissertação. Além disso, apresentamos algumas possibilidades para a exploração em trabalhos futuros a fim de possibilitar a melhoria dos resultados atuais.

Uma das dificuldades que encontramos no início desta pesquisa concerne a falta de bases de dados representativas para a suportar a tarefa de classificação utilizando técnicas de aprendizado profundo. Dada essa limitação, apresentamos uma das contribuições deste trabalho: a criação e disponibilização de uma base de dados composta por vídeos *DeepFake*s de conteúdo adulto. Essa base foi essencial para a continuidade desde trabalho, visto que, por um determinado período, tínhamos acesso apenas ao `FullTIMIT`. Entretanto, ressaltamos que os dados da nossa base podem não conter representatividade para alguns tipos de *DeepFake*s, visto que os sites que utilizamos para a coleta não divulgavam informações referentes a técnica utilizada na produção do conteúdo sintético/falso. Além disso, devido a característica do mercado pornográfico, apenas encontramos vídeos contendo celebridades do sexo feminino.

No decorrer do desenvolvimento deste trabalho novas bases foram sendo disponibilizadas. Inicialmente, as *DeepFake*s se beneficiavam de características presentes em mídias visuais de baixa resolução. Entretanto, as técnicas para criação de faces falsas evoluiu, possando a ser inseridas em contextos de ultra resolução, como no cinema. Devido a isso, realizamos uma análise nos dados de cada base visando identificar características específicas presentes em cada uma delas. Nesta etapa pudemos identificar diferenças entre as técnicas e entender como as *DeepFake*s se comportam em ambientes com variações de iluminação, pose, oclusões, entre outros. Após esta análise, optamos por não utilizar imagens falsas/sintéticas em que foram detectadas mais de uma face, pois não temos um método de distinguir dentre as faces em quais foram aplicadas *DeepFake*s, durante a etapa de extração de faces.

Tendo como base os principais trabalhos de detecção, exploramos quatro abordagens: aprendizado profundo, aumentação de dados, diferentes representações de entrada e função de perda tripla. Buscamos identificar arquiteturas de CNN e as configurações adequadas para a detecção de *DeepFake*s e associa-las com transferência de aprendizado, por meio da utilização de pesos pré-treinados da *ImageNet*. Ressaltamos que, apesar de não conterem características específicas de *DeepFake*s, a diversidade de tipos de imagens que compõem a *ImageNet* auxiliam na extração de características gerais das imagens falsas

além de reduzir o tempo necessário para o treinamento dos modelos. Logo, utilizarmos *ResNet*, *Xception* e *MobileNet* proporcionaram a melhor relação tempo de treinamento x tamanho do modelo x classificação. Além disso, identificamos que uma rede mais simples como a *MobileNet* é suficiente para realizar classificações consistentes.

Adicionalmente, aplicamos técnicas como aumentação de dados e espaços transformados de entrada e percebemos que, para as bases existentes, aplicar tais técnicas não proporcionaram melhorias significativas para a resolução do problema. Acreditamos que isso se deve às características das imagens sintéticas/falsas, que ao serem geradas através das GANs, podem inserir informações úteis para a detecção e que possam estar sendo mascaradas ou alteradas pelo processo de modificação da imagem original.

Ao final dos experimentos com métodos propostos, desenvolvemos um modelo final para cada base de dados. Este modelo foi escolhido, considerando o resultado da acurácia balanceada obtida durante o treinamento, em que alcançamos, para o `FullTIMIT` 100%, `Sensitive` e `FFpp` 99%, e `DFDC` 98%. A fim de nos certificarmos da robustez do modelo desenvolvido, realizamos a predição de um conjunto de imagens do `DFD - Google`, com o qual obtivemos: 60.78% com o modelo `FullTIMIT`, 50.86% com o `Sensitive`, 50% com o `FFpp` e, o modelo mais robusto obteve 85.17% de acurácia balanceada.

Com estes resultados, analisamos individualmente, a classe que foi melhor ou pior classificada. E constatamos que as bases `FullTIMIT` e `Sensitive` não conseguiram classificar satisfatoriamente os exemplos falsos/sintéticos. Acreditamos que isso se deve a características das bases, pois são bases menores e que contém pouca representatividade dos dados. Já para o `FFpp`, apesar de termos classificado corretamente cerca de 99% dos exemplos sintéticos/falsos, obtivemos uma massiva classificação falso-positiva das imagens reais. Isso nos sugere que as diferentes taxas de compressão podem ter sido utilizadas como critério de classificação. Por fim, a representatividade e diversidade com a qual foi composto o `DFDC`, provou ser crucial para generalização do modelo de detecção, frente a diferentes cenários, tendo alcançado 81% de acerto nas imagens falsas/sintéticas e 89% nas reais.

Acreditávamos que combinar as bases pudesse tornar o modelo ainda mais robusto, entretanto, com essa abordagem obtivemos apenas 72.12% de acurácia e analisando a classificação por classe, identificamos que apesar de termos melhorado a classificação dos exemplos falsos/sintéticos em 8 pontos percentuais, regredimos para 55% de acerto para os casos reais. Acreditamos ser necessário realizar experimentos que permitam encontrar a melhor combinação entre as bases de modo a manter o equilíbrio entre a classificação de imagens sintéticas/falsas e reais. Com isso, nossas questões de pesquisa podem ser respondidas.

I. **As bases de dados existentes são suficientes para o desenvolvimento de modelos robustos para detecção de *DeepFake*s?**

   A evolução das técnicas *DeepFake*s associada ao aumento de dados disponíveis, possibilita o treinamento de modelos robustos. Entretanto, de modo a acompanhar a constante inovação das técnicas para *DeepFake*s, serão necessárias atualizações tanto das bases de dados quanto dos modelos para detecção de *DeepFake*s existentes.

II. **É possível gerar um modelo capaz de detectar *DeepFake*s produzidas com**

diferentes técnicas de geração de *DeepFake*s? Em outras palavras, é possível atingir generalização na detecção de tais operações de falsificação, sendo invariante ao modo como a falsificação foi produzida?

É possível, se considerarmos as técnicas para produção de *DeepFake*s atuais. Tendo em vista que, apesar dos modelos produzidos serem robustos para aplicação em diferente técnicas, não é possível assegurar que estes modelos conseguirão detectar *DeepFake*s geradas por técnicas futuras. Idealmente, estes modelos deveriam se manter em constante evolução a fim de acompanhar o estado da arte da criação de *DeepFake*s. Associado a isso, é necessário manter as bases de dados atualizadas de acordo com as novas técnicas que estão em constante desenvolvimento, para que seja possível a criação de modelos mais robustos.

III. **Unir bases de dados construídas com diferentes técnicas para a geração de *DeepFake*s, para produzir um modelo classificador proporciona classificações mais precisas?**

Parcialmente. Ao termos construído um modelo a partir da junção de diversas bases de dados, obtivemos uma classificação mais precisa dos exemplos falsos. Entretanto, o resultado geral não garantiu uma melhoria global, tendo em vista que a precisão em relação aos exemplos reais foi inferior. Com isso, acreditamos que para ser desenvolvido um modelo mais robusto haverá a necessidade de reduzir os casos de falso positivos relacionados aos dados reais. Isso poder ser conseguido com técnicas de adaptação de domínio que buscam aprender as diferenças entre os diversos conjuntos de treinamento objetivando uma maior generalização para o cenário de teste posteriormente.

Ao realizar a análise comparativa entre o nosso método proposto — modelo gerado a partir do treinamento com a base de dados `DFDC` — e as principais técnicas da literatura, durante a predição em um cenário desconhecido (imagens do `DFD - Google`), observamos que o nosso método atingiu o estado-da-arte, estando três pontos percentuais a frente da segunda melhor técnica apresentada na literatura, para a detecção de *DeepFake*s em imagens. Ressaltamos que, apesar de não haver um consenso na literatura quanto à melhor métrica de avaliação para este problema, mantivemos para esta comparação a métrica mais utilizada: AUC.

Acreditamos que com o nosso trabalho, poderemos auxiliar o desenvolvimento das futuras técnicas para detecção, fornecendo algumas informações importantes e, principalmente, orientando os passos iniciais para a continuidade e evolução dos detectores.

- A etapa inicial consiste em selecionar a base de dados que será utilizada, mantendo a solução em concordância com a evolução das *DeepFake*s.
- Identificar algumas de suas principais características, que permita melhor direcionar a solução.
- Dividir os dados para treinamento, evitando contaminar os conjuntos, ou seja, não replicando exemplos de um mesmo indivíduo entre os subconjuntos. Para bases em

que não é possível fazer essa correlação, como no caso do DFDC, sugerimos agrupar todos os dados relativos ao mesmo vídeo.

- Escolher uma arquitetura de rede. Aqui sugerimos utilizar a *MobileNet* versão 1, visto que com ela obtivemos resultados competitivos em cenários intra e inter base de dados, além de ser uma rede leve, o que proporcionará treinamentos mais rápidos e que utilizado pouco espaço para armazenamento.

A partir desses passos iniciais visando a continuidade e aprimoramento da classificação, apresentando alguns aspectos que podem ser explorados.

- Sugerimos realizar o refinamento dos vídeos rotulados como sintéticos/falsos com $\overline{SSIM}$ maior do que o limiar definido nos nossos experimentos (isto é, 0.96). Para isto, poderá ser utilizado o nosso melhor modelo para a predição destes vídeos. Em seguida, os vídeos que foram corretamente classificados podem ser utilizados para incrementar os dados para treinamento e, assim, gerar um novo modelo teoricamente mais robusto.

- Como o foco principal deste trabalho consistiu na detecção de imagens falsas, ao analisarmos os resultados para a predição em cenários desconhecidos, acreditamos ser necessário incrementar também a detecção das imagens reais. Para isso, sugerimos a realização de experimentos entre diferentes bases de dados e o *FFpp*, visto que para esta base, obtivemos os piores resultados para a classificação de imagens reais.

- Acreditamos que associar diferentes bases de dados para produzir um modelo único possa proporcionar um modelo mais robusto. Porém, para que isso de fato ocorra, é necessário identificar e solucionar os problemas que ocasionaram a perda de desempenho durante o teste cego com todas as bases associadas.

- O modelo final atual é composto apenas por uma CNN com transferência de aprendizado. Desse modo, uma possibilidade de melhoria é a aplicação combinada de técnicas como perda tripla, representação de entrada, dentre outras.

- Além disso, acrescentar novas representações de entrada, tais como, Albedo e reflectância [93], podem fornecer informações extras para a rede, e, consequentemente, proporcionar melhores resultados.

- Outra abordagem que acreditamos ser interessante consiste na utilização de *Meta Learning* [131], em que os melhores modelos criados por técnica seriam associados para criar um novo modelo final. Durante o nosso trabalho, gostaríamos de ter explorado esta abordagem, entretanto, considerando os experimentos intra base de dados, o nosso método base proporcionou resultados satisfatórios. Por outro lado, para os experimentos entre diferentes bases, acreditamos ser necessário avaliar outros aspectos relacionados aos dados, antes de partir para a abordagem aplicando *Meta Learning*.

- Finalmente, com a evolução das *DeepFake*s, outras técnicas proposta na literatura, tais como, Imagens híbridas e Cubos temporais [92] utilizados para a definição de novos espaços de entrada e representações de aprendizado espaço-temporais, poderão ser aplicadas.

Referências Bibliográficas

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, e Isao Echizen. Mesonet: a compact facial video forgery detection network. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE, 2018. 20, 26, 28, 34, 36

[2] Henry Ajder, Giorgio Patrini, Francesco Cavalli, e Laurence Cullen. The state of deepfakes: Landscape, threats, and impact. *Amsterdam: Deeptrace*, 2019. 18, 33

[3] Henry Ajder, Giorgio Patrini, e Francesco Cavalli. Automating image abuse: Deepfake bots on telegra. In *Sensity*, 2020. 18, 33

[4] Zahid Akhtar, Dipankar Dasgupta, e Bonny Banerjee. Face authenticity: An overview of face manipulation generation, detection and recognition. In *Nutan College of Engineering & Research, International Conference on Communication and Information Processing (ICCIP)*, 2019. 24

[5] Face Bake. The matrix - he's beginning to believe (nicolas cage) deepfake, 2020. URL `https://bit.ly/35bfCs5`. 17

[6] Sandipan Banerjee, Walter J. Scheirer, Kevin W. Bowyer, e Patrick J. Flynn. Fast face image synthesis with minimal training, 2018. URL `https://bit.ly/3jcj4HY`. 49

[7] Yoshua Bengio, Ian Goodfellow, e Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017. 44

[8] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter Belhumeur, e Shree K Nayar. Face swapping: automatically replacing faces in photographs. In *ACM Transactions on Graphics (TOG)*, volume 27, page 39. ACM, 2008. 24, 27

[9] Branko Blagojevic. How facial recognition works part 2, facial landmarks, 2019. URL `https://bit.ly/3dCYoI1`. 30

[10] Anne Bonner. The complete beginners guide to deep learning, 2019. URL `https://bit.ly/3nZMVXP`. 32

[11] Johnny Botha e Heloise Pieterse. Fake news and deepfakes: A dangerous threat for 21st century information security. In *International Conference on Cyber Warfare and Security*, page 57. Academic Conferences and publishing limited, 2020. 15, 16, 33

[12] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, e Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. *IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pages 67–74, 2018. 47, 58

[13] Ming-Jun Chen e Alan C Bovik. Fast structural similarity index algorithm. *Journal of Real-Time Image Processing*, 6(4):281–287, 2011. 62

[14] Theodore F. Claypoole. Ai and evidence: Let's start to worry. *The National Law Review*, 2019. 33

[15] Momo Company. Zao, 2020. URL `https://bit.ly/3jb3RXy`. 17

[16] Hao Dang, Feng Liu, Joel Stehouwer, Xiaoming Liu, e Anil K Jain. On the detection of digital face manipulation. In *IEEE International Conference Computer Vision and Pattern Recognition (CVPR)*, pages 5781–5790, 2020. 28

[17] Suzanne Day. Mit art installation aims to empower a more discerning public, 2019. URL `https://bit.ly/2H75LvA`. 16

[18] J Deng, W Dong, R Socher, LJ Li, K Li, e L Imagenet Fei-Fei. A large-scale hierarchical image database. In *IEEE International Conference Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 41

[19] Derpfakes. Hillary Clinton | face replacement, 2018. URL `https://bit.ly/35gEmz4`. 16

[20] Derpfakes. Avengers endgame feat. nic cage, 2019. URL `https://bit.ly/2HguehP`. 17

[21] Derpfakes. Nicolas cage | mega mix two, 2019. URL `https://bit.ly/2HguehP`. 17

[22] Tim Dettmers. Deep learning in a nutshell: Core concepts, 2015. URL `https://bit.ly/3dFwnzr`. 30

[23] FaceSwap Dev. Faceswap, 2020. URL `https://faceswap.dev`. 18

[24] Google Developers. Classification: Accuracy, 2020. URL `https://bit.ly/348n08h`. 61

[25] Dimensions. Publications, 2020. URL `https://bit.ly/3pgDbIL`. 20

[26] Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, e Cristian Canton Ferrer. The deepfake detection challenge (dfdc) preview dataset, 2019. URL `https://bit.ly/37mNRj8`. 33, 36, 52

[27] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, e Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 36

[28] Timothy Dozat. Incorporating nesterov momentum into adam.(2016). *Dostupné z: https://stanford.io/3o2Aw5f*, 2016. 42

[29] Nick Dufour e Andrew Gully. Contributing data to deepfake detection research, 2019. URL `https://bit.ly/3m3Wp2a`. 33

[30] Nick Dufour, Google Research, Andrew Gully, e Jigsaw. Contributing data to deepfake detection research, 2019. URL `https://bit.ly/3m3Wp2a`. 33, 36, 52

[31] Ctrl Shift Face. Willem dafoe as hannibal lecter [deepfake], 2019. URL `https://bit.ly/2T99Pxu`. 17

[32] FaceApp. Transform your face using artificial intelligence with just one tap, 2020. URL `https://www.faceapp.com`. 24

[33] FaceSwap. Deepfakes github, 2019. URL `https://bit.ly/2IH6Had`. 51

[34] Paula Fraga-Lamas e Tiago M Fernández-Caramés. Fake news, disinformation, and deepfakes: Leveraging distributed ledger technologies and blockchain to combat digital deception and counterfeit reality. *IT Professional*, 22(2):53–59, 2020. 15, 19

[35] Javier Galbally e Sébastien Marcel. Face anti-spoofing based on general image quality assessment. In *2014 22nd international conference on pattern recognition*, pages 1173–1178. IEEE, 2014. 36

[36] Nils Gehlenborg e Bang Wong. *Heat maps*. Nature Publishing Group, 2012. 27

[37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, e Yoshua Bengio. Generative adversarial networks, 2014. 19

[38] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 42

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, e Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 34

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, e Jian Sun. Deep residual learning for image recognition. In *IEEE International Conference Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 41

[41] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, e Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 68

[42] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, e Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019. 42

[43] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, e Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 41

[44] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, e Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 36

[45] Hyeonseong Jeon, Youngoh Bang, e Simon S Woo. Fdftnet: Facing off fake images using fake detection fine-tuning network. *arXiv preprint arXiv:2001.01265*, 2020. 20, 35, 36

[46] Jigsaw. Creating future - defining technology, 2020. URL `https://jigsaw.google.com`. 52

[47] Kaggle. Deepfake detection challenge, 2020. URL `https://bit.ly/3koINhy1`. 19, 67

[48] Vahid Kazemi e Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014. 30

[49] Ira Kemelmacher-Shlizerman. Transfiguring portraits. *ACM Transactions on Graphics (TOG)*, 35(4):1–8, 2016. 27

[50] Keras. Dense layer, 2020. URL `https://bit.ly/348C4D0`. 62, 101

[51] Keras. Layer activation functions, 2020. URL `https://bit.ly/31lkoCe`. 63, 101

[52] Keras. Flatten layer, 2020. URL `https://bit.ly/2Hm7WLy`. 102

[53] Keras. Globalaveragepooling2d layer, 2020. URL `https://bit.ly/355Ee5m`. 62, 101

[54] Keras. Vgg16 and vgg19, 2020. URL `http://bit.ly/2Y6UYWW`. 36

[55] Keras. Xception, 2020. URL `http://bit.ly/2Y8RzXR`. 36

[56] Keras. Optimizers, 2020. URL `https://bit.ly/3jcQFBm`. 65, 101

[57] Keras. Resnet and resnetv2, 2020. URL `https://bit.ly/3dIua6y`. 36, 63, 101

[58] Rizwan Ahmed Khan, Arthur Crenn, Alexandre Meyer, e Saida Bouakaz. A novel database of children's spontaneous facial expressions (liris-cse). *Image and Vision Computing*, 83:61–69, 2019. 32

[59] Davis E King. Max-margin object detection. *arXiv preprint arXiv:1502.00046*, 2015. 29

[60] Davis E. King. Dlib c++ library, 2020. URL `http://dlib.net`. 29, 58

[61] Diederik P Kingma e Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 42

[62] P. Korshunov e S. Marcel. Deepfakes: a new threat to face recognition? assessment and detection., 2018. 33

[63] Iryna Korshunova, Wenzhe Shi, Joni Dambre, e Lucas Theis. Fast face-swap using convolutional neural networks. In *IEEE International Conference on Computer Vision*, pages 3677–3685, 2017. 27

[64] Marek Kowalski. Faceswap github, 2016. URL `https://bit.ly/35gEPS9`. 51

[65] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, e Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *IEEE International Conference on 3D Vision (3DV)*, pages 239–248. IEEE, 2016. 43

[66] Yann A LeCun, Léon Bottou, Genevieve B Orr, e Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the trade*, pages 9–48. Springer, 2012. 32

[67] Yuezun Li e Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. *arXiv preprint arXiv:1811.00656*, 2018. 20, 34, 36, 82

[68] Yuezun Li, Ming-Ching Chang, e Siwei Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE, 2018. 20, 33, 36

[69] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, e Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics, 2020. URL `https://bit.ly/2T7apMj`. 16, 25

[70] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, e Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *IEEE International Conference Computer Vision and Pattern Recognition (CVPR)*, pages 3207–3216, 2020. 33, 35, 36

[71] Looper. The truth about brandon lee finally revealed, 2020. URL `https://bit.ly/2HfNdZB`. 17

[72] S. Lyu. Deepfake detection: Current challenges and next steps. In *IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, 2020. doi: 10.1109/ICMEW46912.2020.9105991. 33

[73] D. Ma, P. Tang, e L. Zhao. Siftinggan: Generating and sifting labeled samples to improve the remote sensing image scene classification baseline in vitro. *IEEE Geoscience and Remote Sensing Letters*, 16(7):1046–1050, 2019. 42

[74] Laurens van der Maaten e Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 62

[75] Leland McInnes, John Healy, e James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 62

[76] Shervin Minaee. 20 popular machine learning metrics. part 1: Classification and regression evaluation metrics, 2019. URL `https://bit.ly/348zWv0`. 60

[77] Aditya Mishra. Metrics to evaluate your machine learning algorithm, 2018. URL `https://bit.ly/2IH4JXn`. 59, 60

[78] MitFake. Nicolas Cage in Bruce Lee [deepfake], 2020. URL `https://bit.ly/3ja490T`. 17

[79] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 62

[80] J Naruniec, L Helminger, C Schroers, e RM Weber. High-resolution neural face swapping for visual effects. In *Computer Graphics Forum*, volume 39, pages 173–184. Wiley Online Library, 2020. 16, 18, 33

[81] Ryota Natsume, Tatsuya Yatagawa, e Shigeo Morishima. RSGAN: face swapping and editing using face and hair representation in latent spaces. *arXiv preprint arXiv:1804.03447*, 2018. 19, 33

[82] INC. NEOCORTEXT. Zao, 2020. URL `https://reface.app`. 17

[83] Huy H Nguyen, Junichi Yamagishi, e Isao Echizen. Use of a capsule network to detect fake images and videos. *arXiv preprint arXiv:1910.12467*, 2019. 20, 35, 36, 82

[84] Yuval Nirkin, Iacopo Masi, Anh Tran Tuan, Tal Hassner, e Gerard Medioni. On face segmentation, face swapping, and face perception. In *IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 98–105. IEEE, 2018. 27

[85] Yuval Nirkin, Yosi Keller, e Tal Hassner. Fsgan: Subject agnostic face swapping and reenactment. In *IEEE International Conference on Computer Vision*, pages 7184–7193, 2019. 19, 27, 28, 33

[86] Sinno Jialin Pan e Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 41

[87] Shivam B Parikh e Pradeep K Atrey. Media-rich fake news detection: A survey. In *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 436–441. IEEE, 2018. 15

[88] Simon Parkin. The rise of the deepfake and the threat to democracy, 2019. URL `https://bit.ly/31i5T1R`. 16, 33

[89] Ravindra Parmar. Common loss functions in machine learning, 2018. URL `https://bit.ly/3dG28IE`. 44

[90] Sébastien Marcel Pavel Korshunov. Deepfakes: a new threat to face recognition? assessment and detection., 2018. URL `https://bit.ly/2H453PJ`. 20, 26, 33, 36, 50, 57

[91] Ivan Petrov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Jian Jiang, Luis RP, Sheng Zhang, Pingyu Wu, et al. Deepfacelab: A simple, flexible and extensible face swapping framework. *arXiv preprint arXiv:2005.05535*, 2020. 18

[92] Allan Pinto, Helio Pedrini, William Robson Schwartz, e Anderson Rocha. Face spoofing detection through visual codebooks of spectral temporal cubes. *IEEE Transactions on Image Processing*, 24(12):4726–4740, 2015. 90

[93] Allan Pinto, Siome Goldenstein, Alexandre Ferreira, Tiago Carvalho, Helio Pedrini, e Anderson Rocha. Leveraging shape, reflectance and albedo from shading for face presentation attack detection. *IEEE Transactions on Information Forensics and Security*, 15:3347–3358, 2020. 43, 89

[94] Thales Pomari, Guillherme Ruppert, Edmar Rezende, Anderson Rocha, e Tiago Carvalho. Image splicing detection through illumination inconsistencies and deep learning. In *IEEE International Conference on Image Processing (ICIP)*, pages 3788–3792. IEEE, 2018. 43

[95] Jörg Reichardt, Roberto Alamino, e David Saad. The interplay between microscopic and mesoscopic structures in complex networks. *PloS one*, 6(8):e21282, 2011. 34

[96] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, e Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001. 19

[97] Adrian Rosebrock. Facial landmarks with dlib, opencv, and python, 2017. URL `https://bit.ly/3o4NPSS`. 30, 40

[98] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, e Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces, 2018. URL `https://bit.ly/3o5PV52`. 28, 33

[99] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, e Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–11, 2019. 16, 20, 26, 29, 34, 36, 82

[100] Joshua Rothkopf. Deepfake technology enters the documentary world, 2020. URL `https://nyti.ms/2ILJdAT`. 16, 33

[101] Sara Sabour, Nicholas Frosst, e Geoffrey E Hinton. Dynamic routing between capsules. *Advances in neural information processing systems*, 30:3856–3866, 2017. 36

[102] Conrad Sanderson e Brian C Lovell. Multi-region probabilistic histograms for robust and scalable identity inference. In *International Conference on Biometrics*, pages 199–208. Springer, 2009. 36, 50

[103] SANDY SCHAEFER. How furious 7 terminou as cenas de paul walker após sua morte, 2020. URL `https://bit.ly/37kDg8f`. 17

[104] Florian Schroff, Dmitry Kalenichenko, e James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE International Conference Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. 34, 36, 45

[105] scikit learn. sklearn.neural_network.mlpclassifier, 2020. URL `https://bit.ly/2T7908v`. 41, 101

[106] scikit learn. sklearn.svm.svc, 2020. URL `https://bit.ly/3kaZpJG`. 29, 36, 101

[107] Greg Shapiro. Trump resigns - a deepfake prediction | greg shapiro's united states of europe, 2020. URL `https://bit.ly/3dEgVne`. 16

[108] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, e Yan Liu. Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(3):1–42, 2019. 15

[109] Connor Shorten e Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019. 42

[110] Joan E. Solsman. A deepfake bot on telegram is violating women by forging nudes from regular pics, 2020. URL `https://cnet.co/3jloobY`. 18, 19

[111] Fengyi Song, Xiaoyang Tan, Xue Liu, e Songcan Chen. Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients. *Pattern Recognition*, 47(9):2825–2838, 2014. 36

[112] Sander Soo. Object detection using haar-cascade classifier. *Institute of Computer Science, University of Tartu*, pages 1–12, 2014. 29

[113] Dan stacklikemind. Official deepnude algorithm, 2018. URL `https://bit.ly/2T5RfX0`. 18

[114] Kaiming He; Xiangyu Zhang; Shaoqing Ren; Jian Sun. Deep residual learning for image recognition, 2016. URL `https://bit.ly/2H497iX`. 32

[115] Ilya Sutskever, James Martens, George Dahl, e Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013. 42

[116] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, e Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 36

[117] Shahroz Tariq, Sangyup Lee, Hoyoung Kim, Youjin Shin, e Simon S Woo. Detecting both machine and human created fake face images in the wild. In *Proceedings of the 2nd international workshop on multimedia privacy and security*, pages 81–87, 2018. 36

[118] Shahroz Tariq, Sangyup Lee, Hoyoung Kim, Youjin Shin, e Simon S Woo. Gan is a friend or foe? a framework to detect various fake face images. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1296–1303, 2019. 36

[119] Face++ Team. What is face++ ai open platform?, 2020. `https://bit.ly/2IKkInH`. 57

[120] FFmpeg team. Ffmpeg, 2020. URL `https://ffmpeg.org`. 55

[121] OpenCV team. Open source computer vision library, 2020. URL `https://opencv.org`. 56

[122] TensorFlow. tf.keras.optimizers.rmsprop, 2020. URL `https://bit.ly/2T76vD9`. 101

[123] TensorFlow. Tensorflow addons losses: Triplet semi hard loss, 2020. URL `https://bit.ly/3o2Q60N`. 46

[124] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, e Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos, 2016. URL `https://bit.ly/37jC8ll`. 51

[125] Justus Thies, Michael Zollhofer, e Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures, 2019. URL `https://bit.ly/3o96rkO`. 51

[126] Brendan Tierney. Understanding, building and using neural network machine leaning models using oracle 18c, 2020. URL `https://bit.ly/3k7jGQp`. 31

[127] Ruben Tolosana, Sergio Romero-Tapiador, Julian Fierrez, e Ruben Vera-Rodriguez. Deepfakes evolution: Analysis of facial regions and fake detection performance. *arXiv preprint arXiv:2004.07532*, 2020. 20, 35, 36

[128] Phuc Truong. Loss functions: Why, what, where or when?, 2019. URL `https://bit.ly/3jcsYcv`. 44

[129] Usersub. Nick cage deepfakes movie compilation, 2018. URL `https://bit.ly/3lY5BVW`. 17

[130] Vanceagher. Nicholas cage in jumanji (deepfake), 2020. URL `https://bit.ly/2HguWLY`. 17

[131] Ricardo Vilalta e Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002. 89

[132] P Viola e M Jones. Rapid object detection using a boosted cascade of simple features'. In *'IEEE International Conference Computer Vision and Pattern Recognition (CVPR)'*. Hawaii, 2001. 29

[133] Zhou Wang, Eero P Simoncelli, e Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402. Ieee, 2003. 62

[134] Xin Yang, Yuezun Li, e Siwei Lyu. Exposing deep fakes using inconsistent head poses. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8261–8265. IEEE, 2019. 20, 26, 34, 36

[135] Chia-Mu Yu, Ching-Tang Chang, e Yen-Wu Ti. Detecting deepfake-forged contents with separable convolutional neural network and image segmentation. *arXiv preprint arXiv:1912.12184*, 2019. 24

[136] Ying Zhang, Lilei Zheng, e Vrizlynn LL Thing. Automated face swapping and its detection. In *IEEE International Conference on Signal and Image Processing (ICSIP)*, pages 15–19. IEEE, 2017. 27

[137] Peng Zhou, Xintong Han, Vlad I Morariu, e Larry S Davis. Two-stream neural networks for tampered face detection. In *IEEE International Conference Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1831–1839. IEEE, 2017. 20, 34, 36

[138] Ruixi Zhu, Li Yan, Nan Mo, e Yi Liu. Semi-supervised center-based discriminative adversarial learning for cross-domain scene-level land-cover classification of aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 155:72–89, 2019. 45, 46

Apêndice A

Experimentos Completos

Nessa apêndice apresentamos os experimentos realizados, assim como detalhes de parâmetros e configurações, complementando o Capítulo 6.

A.1 `FullTimit`

Iniciamos nossos experimentos utilizando o conjunto de dados `FullTIMIT` (ver Seção 5.3). Para esta base, realizamos nossos experimentos com os classificadores *SVM* [106] e *MLP-Classifier* [105] e a rede *ResNet50*[57] pré-treinada com o *ImageNet* para extração de características (experimentos 1 até o 3).

Para os experimentos com as CNNs, alteramos parâmetros como a taxa de aprendizado (entre 0.1 à 0.0001) e otimizador (SGD [56] e RMSprop [122]), além de diferentes configurações para o topo da rede. Todos esses experimentos foram realizados com *batch* de tamanho 16 e com 4000 exemplos para cada classe (isto é, todos os exemplos desta base de dados).

A seguir, apresentamos a descrição de cada experimento e, na Tabela A.1, os resultados obtidos considerando como métrica o F_1-*Score*.

Experimento 1: Utilizamos o *SVM* [106] com todos parâmetros com seus valores padrões.

Experimento 2: Alteramos o classificador para *MLPClassifier* [105] com todos parâmetros com seus valores padrões.

Experimento 3: Como o resultado do Experimento 3, com métodos tradicionais, foi satisfatório, nesse experimento, já introduzimos a utilização da CNN *ResNet50*[1]. Para tanto, utilizamos uma taxa de aprendizado 0.1, otimizador *RMSprop* e o topo da rede sendo composto pelas camadas *GlobalAveragePolling* [53] e uma camada densa [50] com função de ativação *softmax*[51].

Experimento 4: Como o resultado com essa configuração da *ResNet* não foi bom, para os próximos experimentos, ajustamos os hiper-parâmetros da CNN. O primeiro deles foi com a mesma configuração do experimento 4, apenas alterando a taxa de aprendizado de 0.1 para 0.01.

[1] As características especificas desta arquitetura podem ser conferidas em *resnet2020keras*

Experimento 5: Mantivemos as configurações do experimento 5 e alteramos a taxa de aprendizado para 0.0001. Além disso, trocamos a camada de *GlobalAveragePolling* por uma *Flatten* [52] na saída da rede.

Experimento 6: Retornamos ao experimento 4 e alteramos novamente a taxa de aprendizado de 0.01 para 0.001.

Experimento	F_1**-Score %**
1	51
2	55
3	51
4	91
5	92
6	100

Tabela A.1: F_1-*Score* para cada experimento realizado. Em destaque, a melhor configuração para esta base de dados.

A.2 Face Forensics Plus Plus

A seguir, relatamos as especificações de cada experimento e, na Tabela A.2, informamos o F_1-*Score* obtido. Nos primeiros experimentos, a classe sintética/falsa foi composta por dados de ambas as técnicas.

Experimento 1: Iniciamos utilizando a configuração definida na Subseção 6.3.1.

Experimento 2: Em seguida, alteramos o topo da rede, trocando a camada de *GlobalAveragePolling* para *Flatten*.

Experimento 3: Retornamos às configurações do Experimento 1 e alteramos o otimizador de *RMSprop* para *SGD* com taxa de aprendizado de 0.001.

Experimento 4: Mantivemos a configuração do Experimento 3 e realizamos o treinamento apenas com as imagens *DeepFake*.

Experimento 5: Repetimos o experimento 4, entretanto, apenas com as imagens *faceSwap*.

Experimento 6: Como não obtivemos melhorias, retornamos às configurações do Experimento 1 e alteramos o otimizador para *Adam*, com os parâmetros padrão e taxa de aprendizado de 0.001.

Experimento 7: Reduzimos a taxa de aprendizado de 0.001 para 0.0001.

Experimento 8: Treinamos com a configuração do Experimento 7 apenas em imagens com *DeepFake*.

Experimento 9: Treinamos novamente com a configuração do Experimento 7 apenas em imagens com *faceSwap*.

Experimento	F_1**-Score %**
1	93
2	93
3	93
4	93
5	89
6	33
7	99
8	97
9	96

Tabela A.2: F_1-*Score* para cada experimento realizado. Em destaque, a melhor configuração para esta base de dados.

A.3 Experimentos com o DFDC

Descreveremos nesta seção, os experimentos relativos as camadas do topo da rede (Subseção A.3.1) e perda tripla (Subseção A.3.2) para a base de dados DFDC.

A.3.1 Camadas do Topo da Rede para o DFDC

Detalharemos as camadas adicionadas por experimento, conforme apresentado a seguir, sendo que os resultados obtidos foram descritos na Tabela A.3.

Experimento 1: Iniciamos conforme os experimentos com as demais bases, mantendo uma camada *GAP* seguida de uma camada densa com dois neurônios e ativação *softmax*.

Experimento 2: Em seguida, adicionamos uma camada densa com 256 neurônios e ativação *ReLU* e uma camada densa com dois neurônios e ativação *softmax*.

Experimento 3: Mantivemos a *GAP*, adicionamos uma camada densa com 512 neurônios e ativação *ReLU* e uma camada densa de dois neurônios e ativação *softmax*.

Experimento 4: Associamos a uma camada densa com 512 neurônios e ativação *ReLU*, com uma camada densa com 256 neurônios também com ativação *ReLU* e finalizamos com a uma camada densa com dois neurônios e ativação *softmax*.

Experimento 5: Como obtivemos um resultado melhor com essa configuração a mantivemos e alteramos a camada *GAP* para uma *Flatten*.

Experimento 6: Decidimos diminuir suavemente as informações entre as camadas. Para isso, utilizamos uma camada densa com 512, 256, 128, 64 32, 16, 8 e 4 neurônios, respectivamente, todas com ativação *ReLU* e finalizamos com uma camada densa com dois neurônios e ativação *softmax*.

Experimento 7: Observamos que acrescentar mais camadas não melhorou a classificação, então, reduzimos para uma camada densa com 512 neurônios e ativação *ReLU* e zeramos as ativações de 50% dos neurônios, finalizando com uma *Dense(2) softmax*.

Experimento 8: Como utilizar apenas duas camadas e zerar algumas ativações produziu um resultado pior, adicionamos uma camada densa com 256 neurônios e ativação *ReLU* antes da camada dense com dois neurônios e ativação *softmax*.

Experimento 9: Verificamos que diminuir os dados entre as duas primeiras camadas proporcionou um resultado ainda pior. Então, após a camada densa com 256 neurônios e ativação *ReLU* zeramos a ativação de 50% dos neurônios e finalizamos com a camada densa com dois neurônios e ativação *softmax*.

Experimento	Acurácia (%)
1	88.04
2	50.12
3	74.22
4	79.83
5	85.27
6	85.01
7	82.69
9	79.84
10	84.54

Tabela A.3: Experimentos realizados para definir a melhor configuração para o topo da rede. Em destaque, o melhor resultado obtido.

A.3.2 Perda Tripla para o DFDC

Os detalhamentos dos experimentos realizados são descritos a seguir, e os resultados obtidos estão registrados na Tabela A.4.

Experimento 1: Iniciamos nossos experimentos definindo que 40% da rede A serão mantidos descongelada. Em seguida, alteramos as camadas dessa rede, acrescentando uma camada densa com 1024 neurônios sem ativação, seguida de uma camada com normalização L2 — essa normalização é essencial para o funcionamento da função de perda tripla e estará presente em todos os experimentos desta seção. E para a rede B, apenas uma camada densa com um neurônio e ativação *sigmoid*, sendo esta a última camada para essa rede em todos os experimentos.

Experimento 2: Em seguida, adicionamos uma camada densa com 512 neurônios sem ativação. Enquanto para a rede B, uma camada densa com 512 neurônios e uma camada densa 256 ambas com ativação *ReLU* e uma camada densa com um neurônio e ativação *sigmoid*.

Experimento 3: Na rede A, mudamos para uma camada densa com 1024 neurônios e ativação *ReLU*, considerando 50% de suas ativações, adicionada uma camada densa de 512 neurônios sem ativação. Já para a rede B, desconsideramos 50% das ligações entre os neurônios, seguimos para uma camada densa com 256 neurônios e ativação *ReLU* e a finalização padrão para esta rede.

Experimento 4: Alteramos a estrutura anterior sendo, para a rede A, zeradas as ativações de 50% dos neurônios, seguido por uma camada densa de 256 neurônios sem ativação. Já à rede B, inserimos uma camada densa com 256 neurônios e uma densa com 128 neurônios ambas com ativação *ReLU*.

Experimento 5: Utilizamos para complementar a rede A, uma camada densa com 512

neurônios sem ativação. E, para a rede B, uma camada dense também com 512 neurônios e ativação *ReLU* seguida da finalização anteriormente definida.

Experimento 6: Mesma configuração do Experimento 5 para a rede A. E para a rede B, alteramos a camada densa de 512 para 256 neurônios mantendo a ativação *ReLU*.

Experimento 7: Mesma configuração do Experimento 5 para a rede A. Enquanto que para a rede B, apenas uma *Dense(1) sigmoid*.

Experimento 8: Analisamos a *loss* da rede A e acreditamos ser possível obter um valor ainda inferior. Por isso, zeramos as ativações de 50% dos neurônios após a camada densa com 1024 neurônios e ativação *ReLU* e mantivemos a camada densa de 512 neurônios sem ativação. Já para a rede B mantivemos apenas a sua camada de finalização.

Experimento 9: Mantivemos a configuração final como: Rede A, uma camada densa de 1024 neurônios e ativação *ReLU*, considerando 50% das ativações seguida por uma camada densa com 512 sem ativação. Rede B, uma camada densa com um neurônio e ativação *sigmoid*. Então alteramos a taxa de congelamento da rede A para 50%.

Experimento 10: Mantivemos a configuração do experimento 9 e alteramos a taxa de congelamento da rede A para 25%.

Experimento 11: Por fim, alteramos a taxa de congelamento da rede A, mantendo-a totalmente descongelada.

Experimento	**Rede A loss**	**Rede B acc (%)**
1	0.9916	74.30
2	0.9195	80.26
3	0.8664	80.44
4	0.9797	71.31
5	0.9194	81.52
6	0.9379	80.47
7	0.9149	82.52
8	0.8947	80.88
9	0.9188	86.45
10	0.9320	77.98
11	0.9027	88.09

Tabela A.4: Resumo dos experimentos realizados com a perda tripla. Em destaque a o resultado da melhor acurácia.