

IA Para Detecção de Morphs Faciais

Tasso Eliézer Daflon Cicarino Canellas

IA 1s-2024

UNIFESP

São José dos Campos, Brasil

tasso.canellas@unifesp.br

Abstract—This paper evaluates various algorithms for generating and detecting facial morphing attacks, which pose risks to biometric systems. We assess the effectiveness of OpenCV, MipGAN2, StyleGAN2, and FaceMorpher in creating morphs and use the VGG16 model to classify images as genuine or morphs. Our results show that VGG16 achieves a high accuracy of 98.78% in identifying morphs, outperforming other models such as ResNet 18 and Inception V3. This demonstrates the VGG16 model's strong performance in enhancing biometric security against morphing attacks.

Index Terms—Morphing Attacks, Image Processing , Biometric Security

I. INTRODUÇÃO

Os ataques de morphing representam uma ameaça para sistemas biométricos. Nesse tipo de ataque, um invasor cria uma imagem híbrida combinando características de duas faces diferentes. Essa imagem híbrida é então usada para enganar o sistema biométrico, permitindo que o invasor se passe por outra pessoa.

Quando um indivíduo se apresenta ao sistema biométrico para autenticação, a imagem facial é comparada com os dados armazenados no banco de dados. Se a imagem apresentada for uma combinação de características de várias faces diferentes (um morph), o sistema pode ser enganado. Há também a possibilidade do morph ter sido injetado no banco de dados de referência do sistema biométrico

Esses ataques são especialmente problemáticos em cenários como segurança de fronteiras ou controle de acesso, por exemplo, onde a confiabilidade da identificação é crucial. Embora a pesquisa esteja avançando na detecção de morphs faciais, ainda há poucos conjuntos de dados e ferramentas de detecção de código aberto disponíveis publicamente para combater esse tipo de ameaça.

Para mitigar os riscos associados aos ataques de morphing, é essencial desenvolver modelos de classificação robustos que possam identificar e diferenciar imagens morph de imagens autênticas. Esses modelos precisam ser treinados com grandes conjuntos de dados que incluam uma variedade de morphs e imagens reais para garantir alta precisão e confiabilidade.

A criação de um modelo de classificação eficaz envolve várias etapas, incluindo a coleta de dados, a seleção de características relevantes e o treinamento de algoritmos de aprendizado de máquina. Além disso, é crucial que esses modelos sejam continuamente atualizados e aprimorados para acompanhar as novas técnicas de morphing que surgem.

II. CONCEITOS FUNDAMENTAIS E TRABALHOS RELACIONADOS

A. Morphing attacks

Morphing attacks [1] são uma classe de ataques que visam enganar sistemas de reconhecimento facial. Esses ataques permitem que duas pessoas se registrem no sistema sob a mesma identidade, fornecendo uma imagem falsa que combina os rostos das duas pessoas (o chamado “morph”) como a imagem de referência no momento do registro. Ou também que essa imagem falsa seja apresentada ao sistema e seja autenticada como um indivíduo.

B. GANs

As Generative Adversarial Networks (GANs) [2] são modelos de inteligência artificial que consistem em um gerador e um discriminador. O gerador cria conteúdo (como imagens) e o discriminador avalia se esse conteúdo é real ou falso. As GANs são usadas para gerar imagens realistas, remover ruídos, identificar fraudes e muito mais. Elas têm aplicações em áreas como arte, medicina e design.

C. “Are GAN-based Morphs Threatening Face Recognition?” [3]

O artigo aborda a ameaça representada pelos ataques de morphing em sistemas biométricos de reconhecimento facial.

D. “Face Morphing Attack Generation and Detection: A Comprehensive Survey” [4]

O artigo aborda a vulnerabilidade dos sistemas de reconhecimento facial a ataques de morphing. Ele fornece uma visão sistemática do progresso feito na área de morphing facial, incluindo geração e detecção de morphs, bem como desafios e futuras direções.

E. “A Style-Based Generator Architecture for Generative Adversarial Networks” [5]

O artigo introduz uma nova arquitetura de GANs: StyleGAN, que utiliza um gerador baseado em estilo, separando características de alto nível (como a pose) das variações estocásticas (como texturas). Isso permite um controle mais preciso sobre a geração de imagens. A arquitetura também permite misturar estilos de diferentes imagens, gerando novas combinações. Essa abordagem melhora significativamente a qualidade das imagens geradas e possibilita aplicações criativas, como o morphing facial, onde características de diferentes rostos podem ser combinadas de maneira controlada.

F. “Analyzing and Improving the Image Quality of StyleGAN” [6]

O artigo analisa a qualidade das imagens geradas pelo StyleGAN e propõe melhorias para reduzir artefatos visuais. Ele explora ajustes na arquitetura da rede e nos métodos de treinamento, visando aprimorar a fidelidade das imagens geradas. Além disso, o artigo discute como diferentes configurações afetam a geração de imagens realistas, o que é fundamental para aplicações como o morphing facial, onde a precisão e a naturalidade dos detalhes são essenciais.

G. “MIPGAN - Generating Strong and High Quality Morphing Attacks Using Identity Prior Driven GAN” [7]

O artigo propõe uma técnica para gerar ataques de morphing faciais de alta qualidade utilizando Redes Adversariais Generativas (GANs) com orientação por identidade, chamada MIPGAN. Essa abordagem permite a criação de imagens faciais que combinam características de duas pessoas diferentes de maneira realista e convincente, representando um desafio significativo para sistemas de reconhecimento facial. A metodologia se destaca pela precisão e pela dificuldade em detectar as imagens morfadas.

H. “Face Morph Using OpenCV” [8]

Este estudo propõe um algoritmo que utiliza a biblioteca OpenCV para criar morphs fazendo, para cada ponto chave (como nariz, olhos e boca), uma transformação afim entre os pontos das duas imagens depois da triangulação de Delaunay.

I. “FaceMorpher” [9]

Essa biblioteca também cria os morphs através da escolha de pontos-chave (nesse caso os “landmarks”, pontos de referência que descrevem a geometria ou a forma de um objeto) e triangulação de delaunay em cada ponto. Ela faz a interpolação linear entre cada ponto-chave na primeira imagem e seu respectivo na outra imagem. Assim, ela encontra os pontos intermediários entre as imagens e os combina, criando uma sequência de imagens intermediárias.

III. OBJETIVO

Este trabalho tem como objetivo estudar, desenvolver e discutir, com base em trabalhos anteriores da literatura e implementação de arquiteturas de classificação, a possibilidade de classificar morphing attacks utilizando os conhecimentos obtidos na disciplina de IA e nas pesquisas para o projeto. Estudaremos, portanto, para esse trabalho, a implementação de 4 algoritmos diferentes utilizados para criar morphings (OpenCV, MipGan2, StyleGan2 e Facemorpher) e depois como identificá-los.

IV. METODOLOGIA EXPERIMENTAL

A. Bibliotecas

O projeto foi realizado em Python 3.5.6 para a classificação e 3.7.12 para os morphs com as seguintes bibliotecas:

1) *Bob.io* [10]: A biblioteca bob.io é parte do conjunto de ferramentas Bob, desenvolvida pelo Idiap Research Institute, que é amplamente utilizada em projetos de pesquisa em biometria e aprendizado de máquina. A bob.io lida com operações de entrada e saída (I/O) de dados, oferecendo suporte para diferentes formatos de arquivo, especialmente aqueles usados em aplicações de biometria e visão computacional.

2) *OpenCV* [11]: Open Source Computer Vision Library é uma biblioteca de visão computacional e aprendizado de máquina altamente popular e amplamente utilizada. Desenvolvida originalmente pela Intel, agora é mantida por uma grande comunidade de desenvolvedores. Ela oferece uma vasta gama de funções para processamento de imagens, análise de vídeo, e aplicações de visão computacional, incluindo reconhecimento facial, rastreamento de objetos, e detecção de movimento.

3) *Imutils* [12]: Imutils é uma biblioteca Python leve e conveniente que fornece funções auxiliares para tarefas comuns de processamento de imagens, particularmente aquelas que envolvem OpenCV. Ela foi projetada para simplificar muitas operações básicas que, de outra forma, exigiriam várias linhas de código ao usar diretamente o OpenCV.

4) *PyTorch* [13]: É uma das bibliotecas mais populares e amplamente utilizadas para aprendizado profundo (deep learning). Desenvolvida pelo Facebook AI Research (FAIR), oferece uma interface flexível e intuitiva para construção, treinamento e inferência de modelos de redes neurais. É especialmente apreciada pela comunidade de pesquisa devido à sua facilidade de uso, suporte para computação dinâmica em grafos (define-by-run), e forte integração com a comunidade de aprendizado profundo.

5) *TensorFlow* [14]: É uma biblioteca de código aberto para aprendizado de máquina e aprendizado profundo, desenvolvida e mantida pelo Google Brain Team. Ela fornece uma plataforma completa para a construção, treinamento e implementação de modelos de machine learning, especialmente redes neurais profundas. TensorFlow é amplamente utilizado tanto na pesquisa quanto na produção, suportando uma variedade de tarefas, desde simples classificações até a construção de modelos complexos de visão computacional, processamento de linguagem natural (NLP), e mais.

6) *Matplotlib* [15]: Para a criação e manipulação das imagens apresentadas neste documento, utilizei a biblioteca Python ‘matplotlib’. Esta ferramenta foi crucial para a visualização das imagens e para ajustar seus detalhes antes da inclusão no relatório final. A capacidade do ‘matplotlib’ de gerar gráficos e imagens com alta qualidade facilitou o processo de preparação das imagens para apresentação no LaTeX.

B. Base de Dados

Foram escolhidas duas bases de dados, a primeira foi “Face Research Lab London Set” [16] e depois foi adicionada a base Chicago Face Database [17], que é dividida em três bases: CFD, CFD-MR (Expansão “multiracial” de CFD), CFD-INDIA (Expansão com faces de indivíduos de nacionalidade

indiana). A primeira base de dados escolhida é a Face Research Lab London Set. As imagens são de 102 rostos adultos de 1350x1350 pixels em cores em cada pasta. Cada pasta contém uma foto de determinada posição do rosto de cada indivíduo. A segunda base de dados utilizada foi “Chicago Face Database”, O conjunto principal CFD da base de dados consiste em imagens de 597 indivíduos únicos. Eles incluem asiáticos autodeclarados, modelos femininos e masculinos negros, latinos e brancos, recrutados nos Estados Unidos. Já o conjunto de extensão CFD-MR inclui imagens de 88 indivíduos únicos, que se autodeclararam de ancestralidade multirracial. O conjunto CFD-INDIA inclui imagens de 142 indivíduos únicos, recrutados em Delhi, na Índia.

Todos os modelos são representados com expressões faciais neutras e expressões faciais variadas. Ao total, com ambas bases, temos 1645 faces. Essas bases de dados foram escolhidas por terem como características imagens de fundo branco e indivíduos de camisa branca ou cinza, que seriam parcialmente cortadas durante o pré-processamento. Essas características são úteis para que não haja viés na classificação das imagens e para que as futuras metamorfoses não sejam prejudicadas.

C. Pré-Processamento

Temos como objetivo o estudo de morphings, mas, até então, o dataset era composto apenas de imagens de pessoas reais. Logo, era necessária a criação de morphs com os rostos que já faziam parte da base de dados. Para isso foram escolhidas quatro arquiteturas, MipGAN2, StyleGAN2, Facemorpher, e OpenCV, que utilizam também uma variável alpha entre 0 e 1, a qual quantifica a proximidade do morph resultante com a primeira imagem

Utilizando a primeira base de pessoas reais (FaceLab London) e restringindo em apenas expressões faciais neutras, foram criados morphs de pares de imagens combinadas dois a dois nas arquiteturas escolhidas com alpha (variável que mede a proximidade do morph com a primeira imagem do par) variando entre 0.3, 0.5 e 0.7.. Depois de feito os morphs, para processamento das convoluções na rede, as imagens foram redimensionadas para o tamanho 160x160, com rosto centralizado, utilizando a rede MTCNN [18] (desenvolvida para a detecção de rostos em imagens. Ela combina três redes convolucionais para realizar a detecção facial e o alinhamento de faces.).

Primeiramente, as classes foram separadas pela arquitetura utilizada ou como real. Porém, para as classes não ficarem extremamente desbalanceadas, foram escolhidos 1645 metamorfoses aleatoriamente, 137 de cada alpha em cada arquitetura mais uma escolhida aleatoriamente, para compor a classe dos “morphs”. Assim, a classe dos reais estava composta por 1645 imagens reais e a classe dos “morphs”, por 1645 imagens falsas.

Exemplo com imagens 005 e 030 da base, pela arquitetura StyleGAN2 e alpha = 0.5:

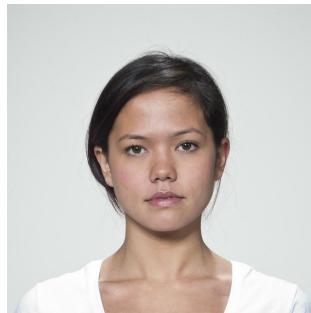


Fig. 1: Indivíduo 30



Fig. 2: Indivíduo 5

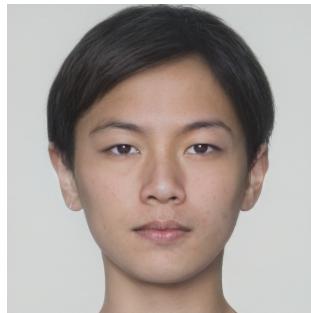


Fig. 3: Morph com StyleGAN2



Fig. 4: Redimensionamento centralizado na face com MTCNN

D. Arquitetura

Foi escolhida a arquitetura VGG16 para a classificação das imagens entre morphs e reais. Uma arquitetura de rede neural convolucional (CNN) desenvolvida pelo Visual Geometry Group da Universidade de Oxford. A arquitetura VGG16 é conhecida por sua simplicidade e eficácia em tarefas de reconhecimento de imagens, utilizando 16 camadas de convolução com filtros de 3x3 e camadas de pooling máximo de 2x2.

Optando pela rede Pré-Treinada, foi utilizada a técnica de Fine Tuning para o treinamento da rede com os novos dados. A rede foi mantida congelada, com exceções das camadas Fully Connected e as últimas três últimas camadas convolutivas.

As últimas camadas foram descongeladas pois morphs faciais geralmente envolvem mudanças sutis em características faciais, como a forma dos olhos, boca, ou o contorno do rosto. As camadas superiores têm a capacidade de capturar essas características complexas e comparar as relações entre elas para identificar anomalias ou mudanças sutis. A rede foi baixada através da biblioteca torchvision.

E. Critérios de Validação

Para o treinamento e avaliação da rede, foram adotados os seguintes critérios:

- Otimizador: Foi utilizado o Gradiente Descendente Estocástico (SGD) como método de otimização. O SGD foi escolhido por sua eficácia em problemas de aprendizado profundo, especialmente em cenários com grandes conjuntos de dados, onde ele realiza atualizações frequentes dos parâmetros com base em lotes menores de dados, o que pode acelerar o processo de convergência.

- Função de Perda: A função de perda selecionada foi a Cross Entropy. A entropia cruzada é amplamente utilizada em problemas de classificação, pois mede a divergência entre as distribuições de probabilidade previstas pelo modelo e as distribuições reais das classes. A minimização dessa função de perda direciona o modelo a melhorar sua capacidade de previsão das classes corretas.
- Métrica de Avaliação: A métrica escolhida para avaliar o desempenho do modelo foi a Acurácia. A acurácia foi utilizada para quantificar a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. Ela é uma métrica intuitiva e amplamente usada, especialmente em cenários onde as classes estão balanceadas.
- Técnica de Validação: Para garantir a robustez e a generalização do modelo, foi empregada a técnica de Cross Validation, mais especificamente o método K-fold com k=5. Nesta abordagem, o conjunto de dados foi dividido em cinco partes (folds), e o modelo foi treinado e validado cinco vezes, cada vez utilizando um fold diferente como conjunto de validação e os restantes como conjunto de treinamento.
- Taxa de Aprendizado (Learning Rate): A taxa de aprendizado utilizada no treinamento do modelo foi de 0,0001. Esse valor controla a magnitude dos ajustes nos pesos da rede neural a cada iteração do otimizador. Uma taxa de aprendizado como essa permite que o modelo faça ajustes consideráveis durante o treinamento, proporcionando um bom equilíbrio entre a velocidade de convergência e a estabilidade.
- Número de Épocas (Epochs): O modelo foi treinado por 200 épocas, o que significa que o conjunto de dados completo foi processado 200 vezes pelo modelo. Um número elevado de épocas permite que o modelo aprenda padrões mais profundos e complexos nos dados, mas também aumenta o risco de overfitting se o modelo começar a "memorizar" os dados de treinamento em vez de generalizar bem para novos dados.
- Salvamento da Melhor Época (Checkpointing): Durante o treinamento, foi implementada uma estratégia de salvamento do modelo na melhor época (best epoch). Isso significa que, ao longo das 200 épocas, o desempenho do modelo foi monitorado, provavelmente usando a métrica de validação, e o estado do modelo foi salvo sempre que essa métrica atingiu seu melhor valor. Essa abordagem garante que, ao final do treinamento, você tenha o modelo com o melhor desempenho possível, evitando a necessidade de escolher manualmente a época ideal e mitigando o risco de overfitting que pode ocorrer nas últimas épocas.

V. RESULTADOS E DISCUSSÕES

A. Checkpoint

Durante o treinamento do modelo, foi marcado o checkpoint para identificar o ponto de menor perda (Loss) durante o processo de validação, apesar de todas as 200 épocas terem sido executadas. Isso nos permitiu não só salvar o ponto de

menor valor de Loss, mas também observar o comportamento das variáveis ao longo das épocas.

Os pontos de checkpoint foram os seguintes:

- Validação 1: Época 137
- Validação 2: Época 111
- Validação 3: Época 118
- Validação 4: Época 97
- Validação 5: Época 111

Observou-se que pequenas variações na taxa de aprendizado não geraram diferenças significativas nos resultados, indicando que o modelo estava suficientemente robusto a essas mudanças.

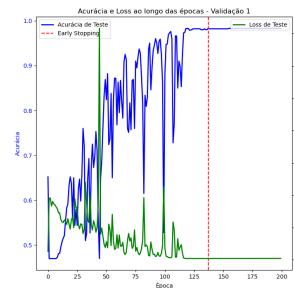


Fig. 5: Validação 1

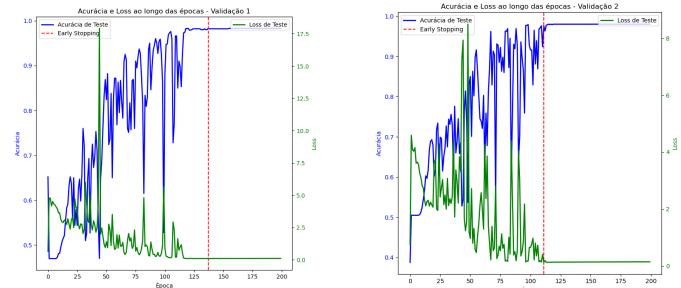


Fig. 6: Validação 2

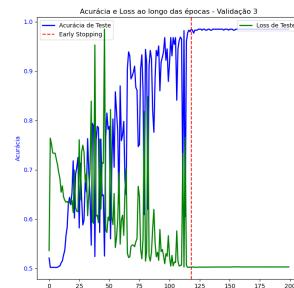


Fig. 7: Validação 3

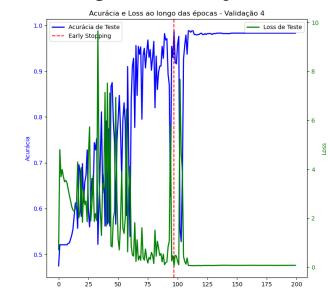


Fig. 8: Validação 4

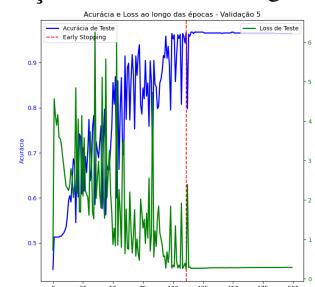


Fig. 9: Validação 5

B. Acurácia na melhor época

A acurácia obtida na melhor época para cada validação foi consistente, com valores altos, demonstrando a eficiência do modelo na tarefa de classificação. Os valores de acurácia foram:

- Validação 1: 98,32%
- Validação 2: 97,56%
- Validação 3: 98,47%

- Validação 4: 98,78%
- Validação 5: 96,79%

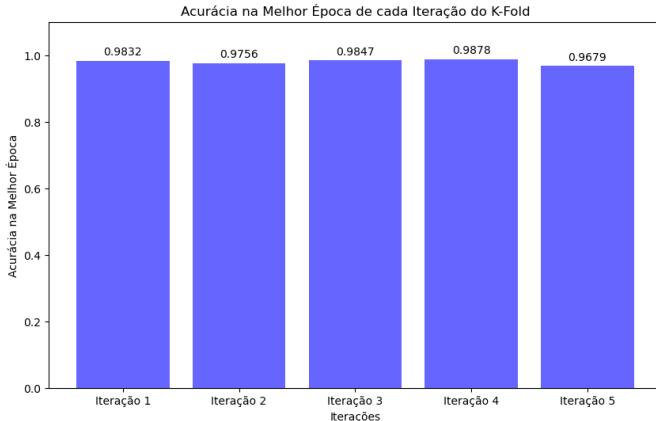


Fig. 10: Acurácia na melhor época das iterações do K-Fold

C. Menor Loss

Os menores valores de Loss durante o treinamento foram:

- Validação 1: 0.0996
- Validação 2: 0.1245
- Validação 3: 0.0907
- Validação 4: 0.0509
- Validação 5: 0.2367

No geral, todos os resultados indicam que o modelo apresentou um bom desempenho, sendo robusto e consistente em suas previsões. A variação observada nos valores de Loss entre as diferentes rodadas de validação pode ser atribuída à forma como os dados foram divididos nos folds do cross-validation. Mesmo com a simplificação dos morphs, é natural que diferentes divisões dos dados levem a pequenas variações nos resultados. No entanto, a consistência geral dos valores de Loss ainda aponta para um modelo que é robusto e bem treinado, apesar da simplificação dos dados.

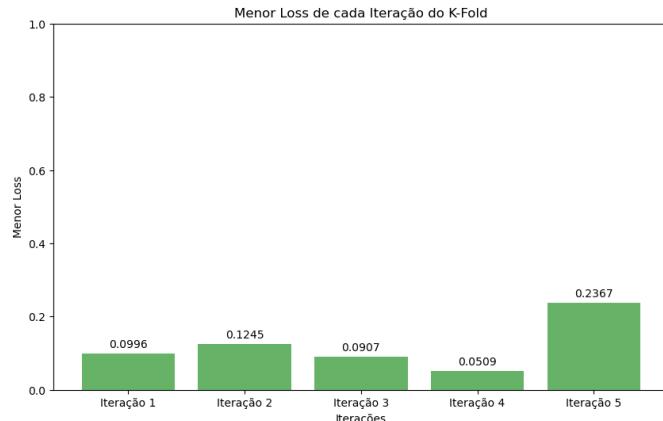


Fig. 11: Menor Loss de iterações do K-Fold

D. FPR x FNR

Os valores de FNR e FPR se mantiveram estáveis ao longo das iterações da validação cruzada, com médias de aproximadamente 0,9% e 2,3%, respectivamente. Esses resultados indicam um equilíbrio entre os falsos positivos e falsos negativos, sugerindo que o modelo não está tendendo para erros específicos de classe.

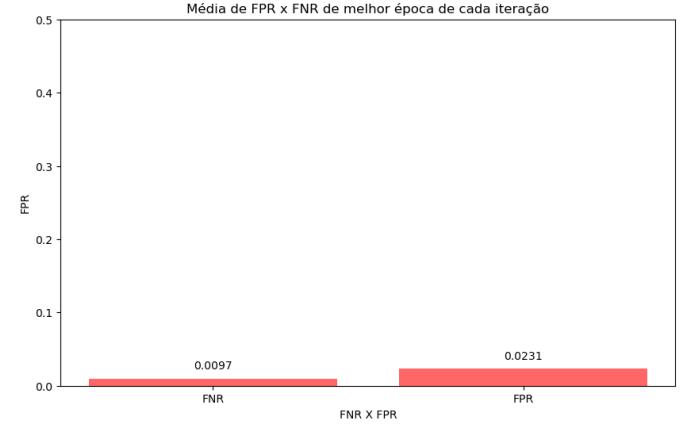


Fig. 12: Média de FPR x FNR de cada iteração

E. Discussões

Para fins de comparação, aplicamos a mesma técnica de treinamento à arquitetura ResNet 18 e à Inception V3 para a classificação das imagens. Os resultados obtidos indicam que a arquitetura ResNet e a Inception V3 não demonstraram um aprendizado consistente, não apresentando melhorias significativas no aprendizado em cada validação do K-fold. Em contraste, a arquitetura VGG16 mostrou-se consistente ao longo das validações, evidenciando um desempenho mais estável.

TABLE I: VGG16 x InceptionV3 x ResNet 18

Table Resultados	Arquitetura		
	ResNet 18	Inception V3	VGG16
Média das melhores acurárias	61,8%	65,04%	97,98%
Média das melhores Loss	1,107	1,334	0,120
Melhor FPR	25,6%	6,68%	2,31%
Melhor FNR	16,5%	27,8%	0,97%

Os resultados obtidos demonstram um desempenho robusto do modelo, com altas taxas de acurácia e baixos valores de Loss, FNR e FPR em todas as rodadas de validação. Esses resultados indicam que o modelo foi eficaz na tarefa de classificação, distinguindo as classes de maneira consistente e precisa. O modelo VGG16 se destacou significativamente, mostrando-se muito mais eficaz do que os modelos Inception V3 e ResNet 18.

Observa-se que a identificação dos morphs pode ter sido facilitada pelas características dos dados, o que contribuiu para a alta performance do modelo. A simplicidade relativa dos morphs utilizados pode ter ajudado na clareza dos padrões

a serem aprendidos, resultando em um aprendizado mais eficiente por parte do modelo.

Enquanto os resultados são promissores, é importante considerar como o modelo se comportaria com morphs mais refinados e realistas, que poderiam reintroduzir artefatos ou nuances mais difíceis de distinguir.

VI. CONCLUSÃO E TRABALHOS FUTUROS

Portanto, este estudo evidencia que o modelo de rede neural foi altamente eficaz na classificação de morphs. Esses resultados não apenas confirmam a eficácia do modelo, mas também demonstram que é viável classificar morphs com alta precisão.

Embora os resultados sejam encorajadores e estabeleçam uma base sólida para futuras pesquisas, é importante reconhecer que este estudo representa um primeiro passo no desenvolvimento do modelo. Trabalhos futuros visam aprimorar a qualidade dos morphs, incorporando maior complexidade aos dados. Esses aprimoramentos não apenas poderão revelar novos desafios, mas também permitirão uma avaliação mais rigorosa das capacidades do modelo.

Assim, os resultados atuais não só confirmam a possibilidade de classificar morphs com sucesso, mas também abrem caminho para o avanço contínuo da pesquisa, onde o refinamento dos dados e do modelo permitirá explorar todo o potencial e enfrentar desafios mais complexos com maior precisão.

REFERENCES

- [1] L. Colbois and S. Marcel, "On the detection of morphing attacks generated by GANs," arXiv, 01-set-2022. [Online]. Disponível em: <http://arxiv.org/abs/2209.00404>. [Acessado: 24-mai-2024].
- [2] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv, 10-jun-2014. [Online]. Disponível em: <http://arxiv.org/abs/1406.2661>. [Acessado: 26-mai-2024].
- [3] E. Sarkar et al., "Are GAN-based morphs threatening face recognition?" arXiv, 05-mai-2022. [Online]. Disponível em: <http://arxiv.org/abs/2205.02496>. [Acessado: 24-mai-2024].
- [4] S. Venkatesh et al., "Face morphing attack generation & detection: A comprehensive survey," arXiv, 04-nov-2020. [Online]. Disponível em: <http://arxiv.org/abs/2011.02045>. [Acessado: 24-mai-2024].
- [5] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," arXiv, 12-dez-2018. [Online]. Disponível em: <https://arxiv.org/abs/1812.04948>. [Acessado: 3-set-2024].
- [6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning - A critical appraisal," arXiv, 10-dez-2019. [Online]. Disponível em: <https://arxiv.org/abs/1912.04958>. [Acessado: 3-set-2024].
- [7] J. Zhao, L. Zhang, and Y. Wang, "MIPGAN - Generating Strong and High Quality Morphing Attacks Using Identity Prior Driven GAN," arXiv, 15-mar-2023. [Online]. Disponível em: <https://arxiv.org/abs/2303.05845>. [Acessado: 3-set-2024].
- [8] S. Mallick, "Face morph using OpenCV — C++ / Python," *LearnOpenCV*. [Online]. Disponível em: <https://learnopencv.com/face-morph-using-opencv-cpp-python/>. [Acessado: 25-mai-2024].
- [9] "Facemorpher," *PyPI*. [Online]. Disponível em: <https://pypi.org/project/facemorpher/>. [Acessado: 25-mai-2024].
- [10] "Bob.io Documentation," *IDiap Research Institute*. [Online]. Disponível em: <https://www.idiap.ch/software/bob/docs/bob/docs/stable/index.html>. [Acessado: 3-set-2024].
- [11] <https://www.idiap.ch/software/bob/docs/bob/docs/stable/index.html>. [Acessado: 3-set-2024]. "OpenCV Documentation," *OpenCV*. [Online]. Disponível em: <https://docs.opencv.org/4.x/index.html>. [Acessado: 3-set-2024].
- [12] "imutils," *GitHub*. [Online]. Disponível em: <https://github.com/PyImageSearch/imutils>. [Acessado: 3-set-2024].
- [13] "PyTorch Documentation," *PyTorch*. [Online]. Disponível em: <https://pytorch.org/docs/stable/index.html>. [Acessado: 3-set-2024].
- [14] "TensorFlow API Documentation," *TensorFlow*. [Online]. Disponível em: https://www.tensorflow.org/api_docs/python/tf. [Acessado: 3-set-2024].
- [15] "Matplotlib API Documentation," *Matplotlib*. [Online]. Disponível em: <https://matplotlib.org/stable/api/index.html>. [Acessado: 3-set-2024].
- [16] L. DeBruine and B. Jones, "Face Research Lab London Set," *figshare*, Dataset, 2017. [Online]. Disponível em: <https://doi.org/10.6084/m9.figshare.5047666.v5>. [Acessado: 26-jul-2024].
- [17] A. Ma, D. Correll, and B. Wittenbrink, "The Chicago Face Database: A Free Stimulus Set of Faces and Norming Data," *Behavior Research Methods*, vol. 47, pp. 1122–1135, 2015.
- [18] I. Paz, "MTCNN: Multi-task Cascaded Convolutional Networks," GitHub, Sep. 2018. [Online]. Available: <https://github.com/ipazc/mtcnn>. [Acessado: 03-Set-2024].