UNIVERSIDADE FEDERAL DE SANTA CATARINA

Detecção e reconhecimento de faces para a anonimização de pessoas em vídeos

Leonardo de Oliveira da Silva

Florianópolis
2023

Leonardo de Oliveira da Silva

Detecção e reconhecimento de faces para a anonimização de pessoas em vídeos

Trabalho de conclusão de curso submetido ao curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação, da Universidade Federal de Santa Catarina.
Orientador: Prof. Alexandre Gonçalves Silva

Florianópolis, julho de 2023

Leonardo de Oliveira da Silva

Detecção e reconhecimento de faces para a anonimização de pessoas em vídeos

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para a obtenção do Título de "Bacharel em Sistemas de Informação" e aprovado em sua forma final pelo Curso de Sistemas de Informação da Universidade Federal de Santa Catarina.

Florianópolis, julho de 2023

Prof. Álvaro Junio Pereira Franco
Coordenador do Curso

Banca Examinadora:

Prof. Alexandre Gonçalves Silva
Orientador

Prof. Elder Rizzon Santos
Membro da banca

Prof. Rafael de Santiago
Membro da banca

Dedico este trabalho aos meus familiares e amigos.

Resumo

Diversas aplicações requerem a anonimização de pessoas presentes em vídeos, seja por questões legais, falta de consentimento do uso de imagem, ou até pessoas menores de idade. Naturalmente, o trabalho de detectar e reconhecer essas pessoas deve ser feito em alguma etapa do processo, e caso seja feito na edição do vídeo, pós gravação, tende a ser custoso e demorado, pois exige o trabalho braçal de um ser humano cortando, quadro a quadro, os rostos encontrados. Com avanços tecnológicos recentes, pode-se aproveitar da inteligência artificial para realizar todas as etapas desse processo: detectar, reconhecer e anonimizar as faces presentes nos vídeos. Essa inovação traz consigo muitas vantagens, como a desalocação de um ser humano e o aumento da velocidade na realização do trabalho, além de permitir que a anonimização seja feita diretamente na gravação, de forma a retirar a necessidade de pós-processamento do vídeo. Nesse contexto, este trabalho utiliza técnicas de detecção e reconhecimento de faces já existentes para aplicar um filtro que anonimize os rostos indesejados e impossibilite o reconhecimento dos mesmos. Para isto, foram revisadas as técnicas normalmente utilizadas neste fim, e foram selecionadas as que melhor se aplicam no contexto deste trabalho, e desenvolveu-se uma ferramenta para realizar a anonimização automática. Por fim, analisou-se sua performance utilizando métrica de acurácia, comparando o resultado de uma rotulação manual com a rotulação automatizada. Ao todo, foram realizados 672 testes, e obteve-se mediana da acurácia de 96,7%.

Palavras-chave: Detecção de faces, Reconhecimento de faces, Anonimização de pessoas, Inteligência Artificial.

Abstract

Several applications require the anonymization of individuals in videos, either for legal reasons, lack of consent for image usage, or even underage individuals. Naturally, the task of detecting and recognizing these individuals needs to be performed at some stage of the process. If it is done during video editing, post-recording, it tends to be costly and time-consuming, as it requires manual labor of a human cutting out the faces frame by frame. With recent technological advancements, it is now possible to leverage artificial intelligence to perform all stages of this process: detecting, recognizing, and anonymizing faces in videos. This innovation brings many advantages, such as eliminating the need for human labor and increasing the speed of the task, while also allowing anonymization to be done directly during recording, eliminating the need for video post-processing. In this context, this work utilizes existing face detection and recognition techniques to apply a filter that anonymizes unwanted faces and prevents their recognition. To achieve this, the techniques commonly used for this purpose were reviewed, and the ones that best suited the context of this work were selected. A tool was developed to perform automatic anonymization. Finally, its performance was analyzed using an accuracy metric by comparing the results of manual labeling with automated labeling. In total, 672 tests were conducted, and a median accuracy of 96.7% was obtained.

Keywords: Face detection, Face recognition, People anonymization, Artificial Intelligence.

Lista de ilustrações

Lista de abreviaturas e siglas

CNN	*Convolutional Neural Network*
DCNN	*Deep Convolutional Neural Network*
ELR	*Empirical Likelihood Ratio*
FPVLS	*Face Pixelation in Video Live Streaming*
HI	*Hidden Identities*
HOG	*Histogram of Oriented Gradients*
IA	Inteligência Artificial
LGPD	Lei Geral de Proteção de Dados
MFPA	*Multi-Face Pixelation Accuracy*
MFPP	*Multi-Face Pixelation Precision*
MLP	Perceptron Multicamadas *(Multilayer Perceptron)*
MMOD	*Maximum-Margin Object Detecto*
MP	*Most Pixelated*
NoP	*Number of People*
OPR	*Over-Pixelation Ratio*
PIAP	*Positioned Incremental Affinity Propagation*
PI	*Preserved Identities*
SVM	*Support Vector Machine*
TCC	Trabalho de Conclusão de Curso
UFSC	Universidade Federal de Santa Catarina
YOLO	*You Only Look Once*

Sumário

1 Introdução

A privacidade dos dados pessoais torna-se, cada vez mais, importante no senso comum, e a possibilidade de identificação facial em vídeos não fica de fora. No Brasil, discute-se o tema com mais força desde 2010, quando foi realizada uma importante consulta pública. Desde então, diversas leis surgiram: em 2014 iniciou-se o Marco Civil da Internet, que intensificou na internet o direito à privacidade, e em 2020 entrou em vigor a LGPD (Lei Geral de Proteção de Dados) [Candido, Araújo e Ribeiro 2021].

Ao realizar gravações de vídeo em público, é natural que pessoas indesejadas apareçam. Dessa forma, solicitar a permissão do uso de imagem de cada uma dessas pessoas pode não ser factível, dependendo do contexto da gravação. Portanto, faz-se necessário o uso de alguma edição das gravações para cortar o aparecimento dessas pessoas, ou anonimizá-las. O trabalho manual dessa anonimização usando *softwares* específicos para edição de vídeo tende a ser custoso e demorado, e em trabalhos como o de Grosselfinger, Münch e Arens 2019, conclui-se que pode levar até dez vezes mais tempo do que sua arquitetura automatizada proposta, utilizando IA (Inteligência Artificial).

Durantes as últimas duas décadas surgiram diversos algoritmos baseados em IA para a detecção de faces. As vantagens e desvantagens destes algoritmos normalmente estão relacionadas à precisão e tempo de processamento necessário para sua execução. Dessa forma, as necessidades da aplicação determinam qual algoritmo se adequa mais. Uma transmissão ao vivo requer rápida execução, por exemplo. Dentre os algoritmos pesquisados por Smelyakov et al. 2021, RetinaFace[1] foi o algoritmo com a maior precisão da detecção, porém seu tempo de execução foi um dos maiores. Já na pesquisa de Jadhav et al. 2021, que incluiu outros algoritmos, conclui-se que o algoritmo com maior precisão foi o MTCNN[2].

Detectar faces, dependendo da necessidade de cada aplicação, pode não ser o suficiente. Diversos algoritmos buscam reconhecer cada indivíduo, e assim conseguir diferenciar uma pessoa da outra. Sharif et al. 2017 explora diversas das mais comuns técnicas na área de reconhecimento facial, sendo duas delas Eigenface[3], Gabor Wavelet[4].

Há diversas situações e contextos em que seja necessário anonimizar certas pessoas em um vídeo. Como dito anteriormente, filmagens em locais públicos, filmagens que apareçam pessoas abaixo da maioridade penal, ou até mesmo pessoas que explicitamente não deram permissão para o uso de imagem, são alguns dos exemplos da aplicação de anonimização de faces das quais se anonimiza apenas algumas das faces detectadas, não todas.

---

1 Deng et al. 2020
2 Shams, Nasim et al. 2019
3 Turk 2005
4 Barbu 2010

Este trabalho tem como objetivo desenvolver uma aplicação capaz de detectar e reconhecer faces para anonimizar apenas as pessoas selecionadas, utilizando técnicas precisas, mas não necessariamente rápidas, uma vez que a aplicação não visa sua utilização em vídeos de transmissão ao vivo. Portanto, o tempo de processamento - ainda que importante - não será uma métrica crítica.

1.1 Objetivos

Esta seção apresenta o objetivo geral e objetivos específicos deste trabalho.

1.1.1 Objetivo Geral

Desenvolver um programa que possibilite, através de técnicas de inteligência artificial para detecção e reconhecimento de faces, anonimizar as faces das pessoas selecionadas em um arquivo de vídeo.

1.1.2 Objetivos Específicos

- Analisar as principais técnicas de identificação de faces e reconhecimento de par de faces;
- Desenvolver um algoritmo para a anonimização das faces selecionadas;
- Avaliar a acurácia em comparação com a rotulação manual;
- Tornar público o código-fonte para uso da comunidade interessada.

1.2 Metodologia

Para o desenvolvimento desse trabalho, serão realizadas as etapas descritas a seguir:

1. revisão bibliográfica sistemática em relação às técnicas mais utilizadas e precisas para detecção e anonimização de faces;
2. desenvolvimento de um algoritmo que utilize as técnicas escolhidas;
3. desenvolvimento de um algoritmo que aplique algum filtro nas faces para anonimizá-las;
4. analisar os resultados utilizando a acurácia calculada;
5. redigir as documentações e relatórios necessários para as disciplinas de Introdução ao TCC, TCC I e TCC 2.

1.3 Estrutura do Trabalho

Este trabalho se divide em quatro capítulos, sendo eles:

Capítulo 1, apresenta-se uma breve introdução do problema, os objetivos deste trabalho e a metodologia seguida.

Capítulo 2, apresenta-se a fundamentação teórica e conceitos básicos necessários para o entendimento e implementação da ferramenta. Conceitos estes que serão mencionados no decorrer do trabalho.

Capítulo 3, são analisados dois trabalhos correlatos a este, ao mesmo tempo que traz um comparativo.

Capítulo 4, descreve o processo de implementação da ferramenta proposta neste trabalho, assim como as tecnologias escolhidas.

Capítulo 5, descreve como os testes foram realizados, e analisa o impacto das variáveis nos resultados.

Capítulo 6, apresenta as considerações finais sobre o trabalho e aponta os possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos necessários para o entendimento e desenvolvimento do trabalho. São apresentados alguns dos métodos mais encontrados em trabalhos relacionados à detecção, reconhecimento e anonimização de faces.

2.1 Redes Neurais

2.1.1 Perceptron Multicamadas

Segundo Gardner e Dorling 1998, o Perceptron Multicamadas (em inglês *Multilayer Perceptron*, MLP) é uma rede neural simples, com uma camada de entrada, camadas ocultas - que podem ser uma ou mais, e a camada de saída. MLP é totalmente conectada, significando que todos neurônios de uma camada são conectados na camada posterior. O perceptron multicamadas também tem como característica possuir uma direção de processamento de informações, portanto sendo considerado como uma rede neural *feed-forward*.

2.1.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (em inglês *Convolutional Neural Network*, CNN) são bem parecidas com MLPs, em termos de estrutura. Também possui camadas de entrada, camadas ocultas e uma camada de saída. Uma das diferenças entre elas é de que suas camadas possuem formato bidimensional, ao contrário do MLP, que é unidimensional [Botalb et al. 2018]. As camadas ocultas podem realizar a convolução, a fim de realçar características, ou podem realizar o *pooling*, que reduz a amostra das características extraídas (tornando o processamento mais rápido) [Desai e Shah 2021].

2.2 Detecção de Faces

Para o reconhecimento de uma face, é necessário primeiramente que o algoritmo detecte a existência de uma face na imagem (ou quadro de um vídeo). O *dataset* WIDER FACE [Yang et al. 2016] é utilizado por diversos trabalhos para a realização de *benchmarks* relacionados à detecção de faces. Em sua página de resultados[1], encontram-se diversas submissões de trabalhos e comparações de seus resultados, nas diferentes dificuldades providas pelo *benchmark*.

Segundo Kumar, Kaur e Kumar 2019, há - de maneira geral - duas abordagens para a detecção de faces a partir de uma imagem: abordagem baseada em características e a abordagem

[1] <http://shuoyang1213.me/WIDERFACE/WiderFace_Results.html>

baseada em imagem. A abordagem baseada em características consiste em extrair características da imagem que combinem com as características faciais conhecidas. Por outro lado, a abordagem baseada em imagem tenta obter a melhor correspondência entre imagens de treinamento e de teste. Encontram-se na Figura 1 algumas das diferentes técnicas para a detecção de faces, organizadas de acordo com as abordagens citas anteriormente.

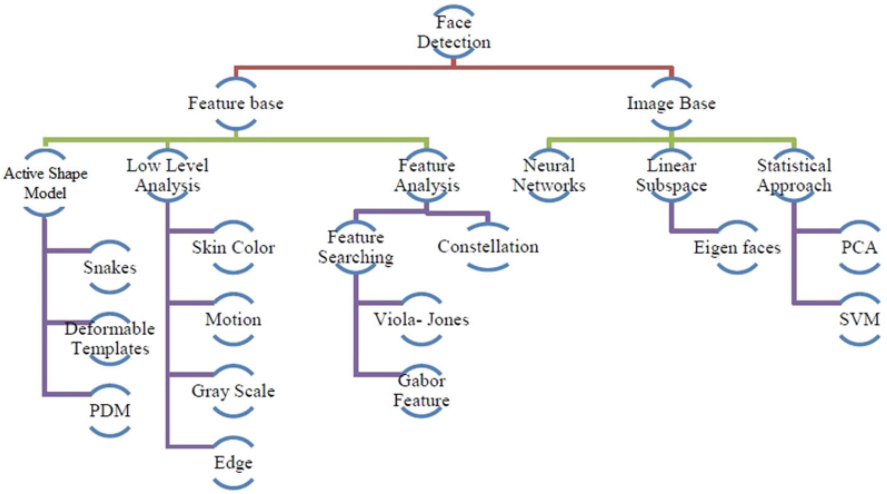


Figura 1 – Diferentes técnicas para a detecção de faces [Kumar, Kaur e Kumar 2019]

Abaixo encontra-se um breve resumo de alguns dos métodos mais relevantes para a detecção de faces em imagens.

2.2.1 Dlib

Dlib é um *toolkit* escrito em C++ para aplicações de aprendizado de máquina e análise de dados. É bastante utilizado na detecção de faces e na detecção de pontos de referência faciais (*landmarks*), pois pode estimar 68 pares de *landmarks* faciais [Tran 2021].

- **Dlib-HOG:** é um método de extração de características usando *Histogram of Oriented Gradients* (HOG) e *Support Vector Machine* (SVM). Esse método não obtem sucesso com poses muito variadas, funcionando apenas com faces frontais, ou levemente rotacionadas [Tran 2021]. Seu funcionamento basicamente se dá pela extração das características para um vetor que servirá de entrada para um algoritmo SVM, que então avaliará se há ou não uma face em determinada região. Essas características extraídas são histogramas das direções dos gradientes da imagem [Jadhav et al. 2021].

- **Dlib-CNN:** é a combinação de uma Rede Neural Convolucional, ou em inglês *Convolutional Neural Network* (CNN) com um *Maximum-Margin Object Detector* (MMOD). CNN é um algoritmo de aprendizado profundo que é usado para analisar imagens dadas como entrada [Jadhav et al. 2021].

2.2.2 YOLO

You Only Look Once (YOLO) é um método para a detecção de objetos utilizando aprendizado profundo, descrito por Redmon et al. 2016. A partir de uma imagem como entrada, gera as regiões que delimitam o objeto detectado, retornando informações como a altura, largura, centro e rótulo, além da confiança de previsão de cada uma das regiões. Faz isso utilizando apenas uma única rede neural [Tran 2021].

- **YOLOv5 e YOLO5Face:** nos anos seguintes à publicação original do YOLO [Redmon et al. 2016], diversas atualizações foram publicadas, uma vez que os autores originais não tiveram interesse em dar continuidade à pesquisa. YOLOv5[2] é a atualização mais recente aceita pela comunidade, apesar de não possuir um artigo publicado [Qi et al. 2021]. Sendo o YOLO um detector de objetos, Qi et al. 2021 realizaram uma série de modificações e desenvolveram o YOLO5Face, com o intuito de especializar para a detecção de faces. Nos testes realizados pelos autores, concluiu-se que tanto o modelo maior (YOLOv5l6) quanto o modelo menor (YOLOv5n) alcançam desempenho próximo ou superior ao estado da arte nos subconjuntos fácil, médio e difícil de validação do WiderFace.

2.3 Reconhecimento de Faces

O reconhecimento de faces consiste em etiquetar cada uma das faces detectadas em uma imagem. No trabalho de Kortli et al. 2020, os métodos de reconhecimento de faces são classificados em três diferentes abordagens: local, holística e híbrida. Essa classificações podem ser observadas na Figura 2.

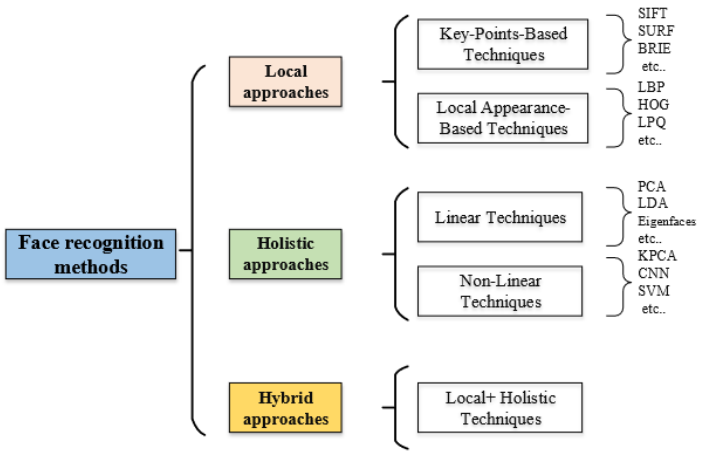


Figura 2 – Classificação de métodos de reconhecimento de faces [Kortli et al. 2020]

Este trabalho apresenta alguns métodos utilizados para o reconhecimento de faces.

[2] <https://github.com/ultralytics/yolov5>

2.3.1 ArcFace

Em sua publicação, Deng et al. 2019 descreve que um dos principais desafios ao utilizar *Deep Convolutional Neural Networks* (DCNNs) para reconhecimento de faces em larga escala é o projeto de uma função de perda apropriada. O trabalho propõe uma função de perda chamada *Additive Angular Margin Loss* (ArcFace), que visa melhorar o poder discriminativo do modelo de reconhecimento facial. Dessa forma, substitui-se a função de perda tradicional *softmax*, uma vez que o tamanho da matriz de transformação linear aumenta linearmente com a quantidade de identidades [Tran 2021].

2.3.2 SphereFace

Liu et al. 2017 introduz uma função de perda *softmax* angular (A-Softmax). O objetivo é obter uma distância intraclasse máxima menor do que a distância interclasse mínima, que permite as redes neurais convolucionais (CNNs) aprender características discriminativas angulares.

2.4 Anonimização de Faces

Há diversas formas de anonimizar faces. A mais simples consiste em embaçar os rostos desejados. Caso seja desejado preservar as características faciais, ao mesmo tempo que a identidade seja anonimizada, é possível utilizar uma abordagem de ofuscação de atributos.

2.4.1 Embaçamento da Face

Pulfer 2019 comparou, em seu estudo, três métodos de processamento de imagem para determinar qual método é mais efetivo em reduzir a possibilidade de detecção de faces. Para tal, o autor implementou cada um dos métodos selecionados, com diferentes parâmetros, e compararam a quantidade de faces detectadas nas imagens processadas com as imagens originais. Os métodos testados foram:

- ***Box Blurring:*** é uma forma de filtro *low-pass*. Utiliza valores uniformes no *kernel*, por isso é relativamente simples de calcular comparado a outros métodos, e portanto mais rápido;
- ***Gaussian Blurring:*** também é uma forma de filtro *low-pass*, porém utiliza valores derivados de uma função Gaussiana;
- ***Differential Privacy Blurring:*** esse método duplica a cor de cada *pixel* nos *pixels* vizinhos, de acordo com o tamanho de bloco desejado, e, além disso, também adiciona um ruído à cor de cada bloco, sendo este ruído aleatório de acordo com a distribuição de Laplace.

Dos três métodos, o *Differential Privacy* foi o mais efetivo em negar a detecção de faces nas imagens processadas. *Box Blurring* foi mais efetivo do que o *Gaussian Blurring*, que teve a menor efetividade dentre os três.

2.4.2 Ofuscação de Atributos

O método proposto por Li et al. 2021 visa, a partir de uma face, gerar uma nova face correspondente. Esta nova face deve possuir uma aparência diferente da original, ainda preservando características relacionadas. O trabalho seleciona partes sensíveis da face, que são chaves para o processo cognitivo de reconhecimento. Estas partes são cabelo, sobrancelhas, olhos, nariz e lábios. O estudo analisa a modificação de diferentes números de atributos. Li et al. 2021 concluem que o método proposto é capaz de preservar a identificação de faces, ao mesmo tempo que anonimiza a aparência facial.

3 Trabalhos Relacionados

Este capítulo apresenta dois trabalhos para a solução de problemas que, de alguma forma, são semelhantes ao abordado neste trabalho. Inicialmente, descreve-se uma tese que desenvolve técnicas variadas para acelerar o reconhecimento das faces em vídeos [Tran 2021]. Na sequência, discorre-se um trabalho que foca na proteção da privacidade em vídeos de transmissão ao vivo [Zhou e Pun 2020].

Para encontrá-los foi utilizada o Google Scholar[1], uma ferramenta para a pesquisa de literatura acadêmica e jornais científicos. Alguns termos utilizados na busca foram *"face detection"*, *"face recognition"* e *"face anonymization"*. Dentre os resultados, foram selecionados os artigos mais relevantes, sendo um dos fatores sua similaridade com a proposta deste trabalho.

3.1 *Fast Video-based Face Recognition in Collaborative Learning Environments*

Este trabalho tem como objetivo desenvolver um método de reconhecimento de faces em vídeo para reconhecer estudantes independente de suas poses. O método desenvolvido utiliza a abordagem *k-means* para identificar agrupamentos de imagens a fim de reconhecer faces em diferentes poses. Depois aplica a otimização multiobjetivo para estudar a interdependência entre a taxa de reconhecimento, o número de *clusters* e a precisão do reconhecimento [Tran 2021].

Este trabalho utiliza o próprio vídeo de entrada para a geração dos protótipos de face de cada pessoa identificada. Protótipos estes que, mais tarde, serão utilizados para reconhecer, quadro a quadro do vídeo, quais estudantes estão interagindo entre si. Buscando acelerar e precisar o reconhecimento das faces, o autor analisou diferentes técnicas (e suas combinações) no processamento das faces detectadas. Ao todo, foram propostos quatro métodos que aplicam as técnicas, e também uma base de referência em que nenhuma dessas técnicas é aplicada. As técnicas analisadas foram:

- ***Data Augmentation*:** em alguns casos, diversas pessoas não aparecem com frequência, ou não se movem o suficiente para a obtenção de poses variadas. Essa técnica tenta resolver esses problemas ao criar, virtualmente, diferentes poses a partir de uma única imagem da face, aumentando então a variedade de poses. Para isso, aplicam-se a combinação de algumas edições na imagem: espelhamento horizontal, rotação, alteração da escala, movimentação horizontal e vertical, e transformação de cisalhamento.

[1] <scholar.google.com/>

- ***K-means Clustering*:** na base de referência, todas as faces de cada participante foram utilizadas para criar uma coleção de protótipos associados com os participantes. Porém, cada um possuía aproximadamente 9.500 imagens, o que levaria muito tempo para processar. Portanto, foi introduzida a técnica *K-means Clustering*. Essa técnica consiste em utilizar o algoritmo *K-means* para agrupar faces muito semelhantes, quando a pessoa não se move muito entre um quadro e outro do vídeo, por exemplo. Dessa forma, o trabalho espera que os centróides resultantes do *K-means* representem as diferentes poses de cada participante.
- ***Sparse Sampling*:** essa técnica consiste em realizar uma amostragem esparsa dos quadros do vídeo, no lugar de utilizar todos os quadros. Por exemplo, o algoritmo apresentado utiliza apenas um quadro por segundo do vídeo para alcançar a escassez.
- ***Frames Skipping*:** para acelerar o processamento, algumas opções de pulos de quadros foram testadas: sem pulos, 5, 10, 15, 20, 30 e 60 quadros.

Para a base de referência, foi treinado e testado um modelo de classificador SVM usando o código disponibilizado pelo scikit-learn[2]. Além deste, outros oito classificadores foram implementados a fim de comparar os resultados entre estes modelos, sendo o SVM obtendo o melhor resultado.

Tendo os resultados para uma base de referência, seguiram-se os testes dos métodos propostos no trabalho. Concluiu-se que o método final e otimizado resultou em um reconhecimento mais rápido e com melhora na acurácia do reconhecimento de faces. O método proposto atingiu uma acurácia de 71,8%, enquanto o modelo de referência teve 62,3%, enquanto rodou mais rápido que todos os outros métodos propostos.

3.2 *Personal Privacy Protection via Irrelevant Faces Tracking and Pixelation in Video Live Streaming*

Neste trabalho, Zhou e Pun 2020 afirmam que com a popularização das plataformas de *streamings* de vídeo, tornou-se preocupante a privacidade das pessoas contidas nos vídeos. Os autores reiteram que é inviável para essas plataformas de *streaming* alocar esforço humano para executar as anonimizações necessárias. Citam que essas plataformas, incluindo YouTube e Microsoft Azure, focam no processamento *offline* dos vídeos, deixando o campo das transmissões ao vivo inexplorado. Portanto, neste artigo apresenta-se um novo método desenvolvido chamado de *Face Pixelation in Video Live Streaming* (FPVLS), que automaticamente anonimiza as pessoas em *streaming de vídeos*.

Segundo os autores, aplicar somente rastreadores de face acarreta em diversos problemas, como desvio do alvo, eficiência computacional, e superpixelização. Portanto, para realizar a

[2] <https://scikit-learn.org/stable/modules/svm.html>

pixelização de maneira rápida e precisa das faces irrelevantes, o FPVLS é organizado em uma estrutura de dois estágios principais. Em quadros individuais, o FPVLS utiliza detecção de rosto baseada em imagem e *Embedding Networks* para produzir vetores de face. No estágio de geração de trajetória, o algoritmo *Positioned Incremental Affinity Propagation* (PIAP) proposto aproveita os vetores de face e informações de posição para associar rapidamente os rostos da mesma pessoa em diferentes quadros. Porém, essas informações de trajetórias possivelmente serão intermitentes e não confiáveis, portanto introduziu-se uma etapa de refinamento de trajetória, mesclando uma rede proposta com um teste entre duas amostras, baseado na estatística *Empirical Likelihood Ratio* (ELR). Então, aplica-se um filtro gaussiano nas trajetórias antes da pixelização final.

No artigo encontra-se uma figura comparativa entre *frames* do vídeo original, *frames* do mesmo vídeo processado pela ferramenta do YouTube, e também processado pelo FPVLS. É possível encontrar os problemas citados pelos autores, como o desvio do alvo e a superpixelização. Também citam métodos que inicialmente tentaram implementar: 3D CNN, MOT, MFT. Porém encontraram os problemas citados anteriormente.

Considerando os prós e contras dos métodos testados, desenvolveram a FPVLS. Junto com um estágio de pré-processamento e um estágio de pós-processamento, o FPVLS possui dois estágios principais: o estágio de geração de trajetórias de face e o estágio de refinamento de trajetória.

Para a realizar a experimentação, foi necessário coletar diversos vídeos de transmissão ao vivo das plataformas YouTube e Facebook, e construir um *dataset*, uma vez que não há disponível nem um *dataset*, nem um *benchmark*. Por ser o primeiro trabalho realizado no campo de pixelização de vídeos de transmissão ao vivo, os autores não encontraram métodos de comparação, portanto utilizaram os potenciais algoritmos e os adaptaram para a tarefa de pixelização de face. Para as métricas, utilizaram as já amplamente aceitas no campo de rastreamento, e propuseram as seguintes métricas para indicar a performance do algoritmo: *Multi-Face Pixelation Accuracy* (MFPA), *Multi-Face Pixelation Precision* (MFPP), e *Most Pixelated* (MP). Além disso, *Over-Pixelation Ratio* (OPR) e a capacidade de lidar com o número de pessoas *Number of People* (NoP) são as outras duas métricas personalizadas para a tarefa de pixelização.

No geral, o FPVLS obtém melhor desempenho em MFPA, MFPP, MP e OPR em todos os subconjuntos de dados. O método proposto aumenta notavelmente a métrica MP em todo o conjunto de dados, indicando que o FPVLS produz uma pixelização mais duradoura, uma vez que esta métrica está relacionada à quantidade de frames consecutivos que foram pixelizados corretamente.

3.3 *An architecture for automatic multimodal video data anonymization to ensure data protection*

Neste trabalho, Grosselfinger, Münch e Arens 2019 projetaram e implementaram um *pipeline* de anonimização de vídeos (anonimização de faces e placas de carro). Este *pipeline* contém *plugins* para ler, modificar e gravar diferentes formatos de imagem, bem como métodos para detectar as regiões que devem ser anonimizadas. Isso inclui um método para determinar as posições da cabeça e um detector de objetos para as placas, ambos baseados em métodos de aprendizado profundo de última geração.

Para a detecção das faces é estimado primeiramente a pose do corpo da pessoa, utilizando OpenPose[3]. O retorno de OpenPose inclui seis pontos da região facial, e a partir desses pontos é possível determinar o centro da face. O tamanho da região facial é determinado a partir das distâncias entre esses pontos.

As regiões a serem anonimizadas definidas são, então, salvas em um arquivo XML, onde é possível realizar algum refinamento manual, caso necessário. Por exemplo, eliminar falso positivos, adicionar ou alterar regiões detectadas. A anonimização é realizada com uma versão modificada do *gaussian blur*, tratado neste trabalho em 2.4.1. Segundo os autores, o processo de anonimização atinge desempenho quase humano, sem precisar de praticamente nenhum trabalho humano.

3.4 Comparativo

Cada um desses trabalhos contribui de alguma forma para a realização deste.

O primeiro trabalho é o que mais se assemelha à proposta deste. Todas as etapas a serem desenvolvidas neste trabalho são, na medida do possível, bastante semelhantes às etapas da ferramenta a ser desenvolvida. A etapa de reconhecimento, porém, se difere um pouco, uma vez que na ferramenta proposta o usuário irá entrar com as imagens para a geração dos protótipos de face.

O segundo trabalho apresenta uma proposta de rastreamento das faces, que poderia ser uma adição bastante interessante para a ferramenta proposta neste trabalho.

O terceiro trabalho realiza a anonimização, além de faces, de placas de carro, o que não é o foco deste trabalho. Porém, utiliza do salvamento das regiões detectadas em um arquivo, permitindo - dessa forma - que o usuário faça ajustes antes do processamento final. Esta também é uma adição a ser considerada para a ferramenta proposta.

[3] Cao et al. 2019

4 Desenvolvimento

Este capítulo apresenta as ferramentas utilizadas e o desenvolvimento do *framework* proposto.

4.1 Premissas e Decisões

Para o desenvolvimento deste projeto, algumas escolhas relacionadas à tecnologias e premissas de funcionamento foram feitas, e encontram-se abaixo:

4.1.1 Entradas e Saídas

Este trabalho parte da premissa que serão fornecidos:

- Uma lista com os nomes das pessoas a serem mantidas;
- Exemplares das faces das pessoas a serem mantidas, sejam eles imagens ou vídeos;
- Vídeo a ser processado.

E terá de saída o arquivo de vídeo com as faces desejadas anonimizadas.

4.1.2 Detecção e Reconhecimento de Faces

Em relação à detecção e reconhecimento de faces, optou-se por utilizar a biblioteca InsightFace. Esta escolha visou a simplificação do desenvolvimento, uma vez que ambas as funções (detecção de face e geração dos *embeddings*) já são implementadas, evitando a necessidade de múltiplas bibliotecas. Será discutida com mais profundidade posteriormente neste trabalho.

4.1.3 Anonimização

Conforme discutido em 2.4, existem diversas formas para anonimizar uma face. Neste trabalho o autor decidiu por realizar um filtro do tipo *Gaussian Blur*. Este filtro é amplamente conhecido, e já se encontra implementado na biblioteca utilizada para manipulação de imagens, OpenCV.

Para os autores deste trabalho, o método de anonimização através da ofuscação de atributos seria o ideal, porém aumentaria consideravelmente a complexidade de implementação. Por outro lado, preencher a região da face a ser anonimizada com *pixels* seria o método mais simples, mas não seria agradável àqueles espectadores do vídeo processado pelo *framework*

proposto. Os autores consideram o *Gaussian Blur* um método que junta as vantagens de cada um desses outros dois métodos.

4.2 Ferramentas

Nesta seção discutem-se algumas das ferramentas utilizadas durante a implementação do trabalho.

4.2.1 InsightFace

InsightFace[1] é uma biblioteca para Python que implementa uma diversidade de algoritmos relacionados à análise de face, como reconhecimento, detecção e alinhamento. Nela é possível encontrar diversos modelos já pré-treinados, que aceleram e facilitam o desenvolvimento do projeto.

4.2.2 Keras e Tensorflow

Keras[2] é uma interface de código aberto que facilita a utilização da biblioteca, também de código aberto, Tensorflow[3]. Com elas pôde-se facilmente implementar redes neurais com estruturas variadas durante o desenvolvimento e testes do projeto.

4.2.3 Scikit-learn

Scikit-learn[4] é uma biblioteca de *machine learning* para Python, e foi utilizada durante o treinamento do MLP desenvolvido.

4.2.4 OpenCV

OpenCV[5] é uma biblioteca para manipulação de imagens, e é desenvolvida principalmente em linguagem C++. Para utilizá-la neste projeto, fez-se o uso da biblioteca *opencv-python*. OpenCV foi utilizada neste projeto com a finalidade de criar um filtro de embaçamento na região da face detectada.

4.3 Projeto do programa

Para fins didáticos este trabalho pode ser abstraído em três etapas:

1 <https://insightface.ai/>
2 <https://keras.io/>
3 <https://www.tensorflow.org/>
4 <https://scikit-learn.org>
5 <https://opencv.org/>

- Aprender a reconhecer as pessoas desejadas;
- Detectar e classificar, quadro a quadro, todas as faces presentes no vídeo;
- Anonimizar as faces desejadas.

Na figura abaixo, encontra-se a relação das três etapas citadas anteriormente, com as três principais classes do código desenvolvido:

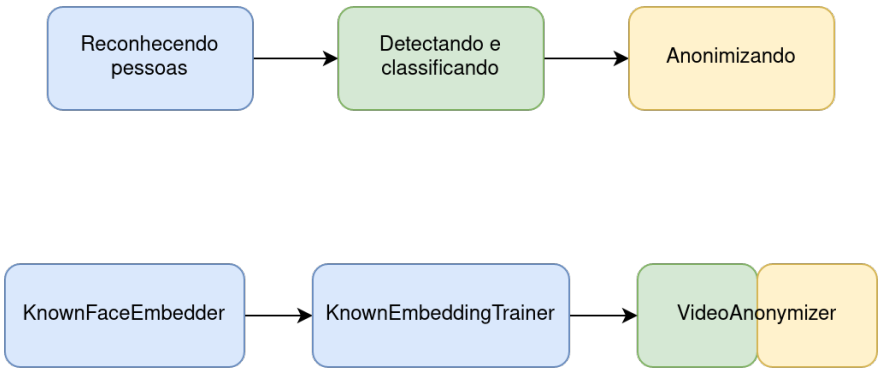


Figura 3 – Abstração do projeto do programa e estrutura do código desenvolvido

Essas etapas são encontradas nas subseções abaixo:

4.3.1 Reconhecendo pessoas

Para cada uma das pessoas fornecidas, gera-se com a InsightFace os *embeddings* de todas as imagens, ou *frames* dos vídeos. Cada um desses *embeddings* é rotulado com o nome da pessoa, e finalmente salva-se o resultado em um arquivo.

A partir dos *embeddings* rotulados treina-se um MLP capaz de classificar - com um grau de confiança atribuído - a quem pertence aquela face.

4.3.2 Detectando e classificando

Com o MLP treinado, dá-se início à segunda etapa, onde são detectadas e classificadas todas as faces presentes no vídeo. Para tal, processa-se um *frame* de cada vez do vídeo, e utiliza-se novamente da InsightFace para detectar as faces e gerar seus respectivos *embeddings*. Então, para cada um desses *embeddings*, alimenta-se o MLP e verifica qual a pessoa identificada (isto é, qual das classes possui a maior confiança).

Enfim, as informações de cada face: região, pessoa identificada e grau de confiança (que vai de 0 a 1), são salvas em um arquivo, e serão utilizadas na próxima etapa.

4.3.3 Anonimizando

Por último, o vídeo é processado novamente do início, utilizando as faces detectadas na etapa anterior. Caso a confiança ultrapasse o limiar configurado, tem-se certeza de que aquela face detectada realmente pertence àquela pessoa.

Caso não haja certeza que a face pertence à pessoa a ser mantida, ou a pessoa identificada não está inclusa na lista de pessoas a serem mantidas, ou seja, deve ser anonimizada, utiliza-se a região detectada para criar um filtro do tipo *Gaussian Blur*.

4.4 Código desenvolvido

A partir da estrutura abstraída do programa, pode-se compreender melhor o código desenvolvido.

4.4.1 Geração dos *embeddings*

A classe desenvolvida responsável pela geração dos *embeddings* foi chamada de **KnownFaceEmbedder**. Tem-se como entrada as imagens que possuem as faces, rotuladas através da estrutura de diretórios que se organizam. A partir disso, para cada imagem é gerado um vetor de *floats*, isto é, o *embedding* da face presente naquela imagem. Então, é realizado o salvamento de um arquivo composto pelo mapeamento de um rótulo com um *embedding*. A Figura 4 apresenta um diagrama das entradas e saídas da classe KnownFaceEmbedder.

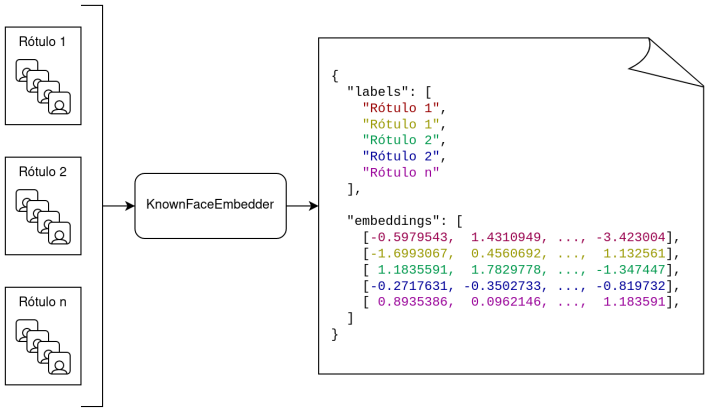


Figura 4 – Funcionamento da classe responsável pela geração dos *embeddings*

4.4.2 Aprendizado

A classe responsável por treinar a rede neural para a classificação das faces foi chamada de **KnownEmbeddingTrainer**. O arquivo gerado contendo os rótulos e os *embeddings* é utilizado

aqui para realizar o treinamento. A Figura 5 apresenta um diagrama das entradas e saídas da classe KnownEmbeddingTrainer.

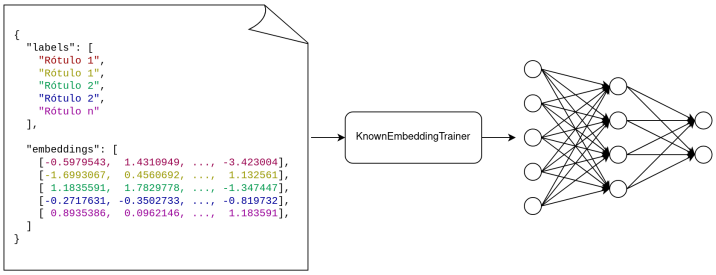


Figura 5 – Funcionamento da classe responsável pela geração dos *embeddings*

4.4.3 Anonimização do vídeo

Por fim, a classe responsável por anonimizar as faces desejadas foi chamada de **VideoAnonymizer**. Para anonimizar o vídeo, esta classe recebe como entrada o arquivo de vídeo a ser processado, o MLP treinado anteriormente, e uma lista de nomes (rótulos) das pessoas a terem sua identidade mantida no vídeo. Aqui também é utilizado o InsightFace para detectar as faces e gerar os *embeddings*, que serão utilizados como entrada do MLP para a classificação de cada uma das faces. Caso a face identificada estiver na lista de nomes, e possuir um grau de confiança maior que o configurado, esta não será anonimizada. Caso contrário, será anonimizada. Essa classe também salva em um arquivo as regiões das faces detectadas, junto com o resultado do MLP (rótulo e probabilidade). A utilização desse arquivo será apresentada na seção 4.5. A Figura 6 apresenta um diagrama das entradas e saídas da classe VideoAnonymizer.

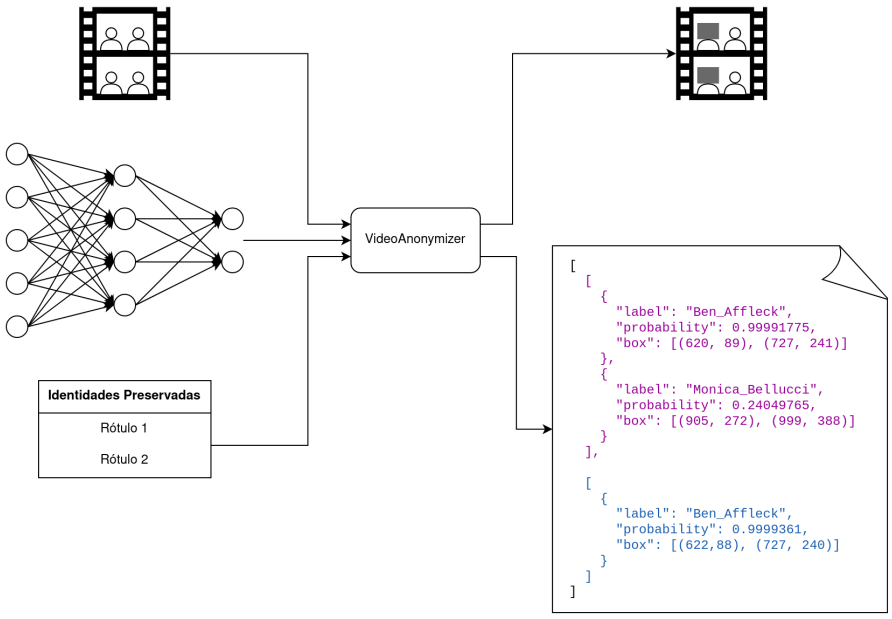


Figura 6 – Funcionamento da classe responsável pela anonimização do vídeo

4.4.4 Uso final do programa pelo usuário

A utilização do programa desenvolvido neste trabalho se dá pela linha de comando do *shell*, ao invocar o Python.

Há um argumento obrigatório, que é o caminho para o arquivo de vídeo que será processado.

As imagens de cada uma das pessoas devem estar dentro de um diretório cujo nome será utilizado para rotulá-las posteriormente. Todos diretórios de pessoas devem estar dentro de outro diretório chamado "datasets". Dois exemplos de caminhos: `./datasets/Ben_Affleck/000001.jpg` e `./datasets/Michelle_Obama/000100.jpg`.

Caso o usuário queira definir o tamanho das camadas do MLP, pode fazê-lo através da *flag* `-l` (ou `--layers`). Se não o fizer, será utilizado o padrão (3 camadas): 256, 256 e 512.

Caso o usuário queira definir as épocas de treino do MLP, pode fazê-lo através da *flag* `-e` (ou `--epochs`). Se não o fizer, será utilizado o padrão: 1.

Naturalmente, também pode ser informado o nome das pessoas que terão suas faces mantidas (não anonimizadas) no vídeo. O usuário o faz através da *flag* `-pi` (ou `--preserved-identities`).

Um exemplo de utilização, em que se mantém apenas as faces do Ben Affleck:

```
python anonimify.py oscar.mp4 -pi Ben_Affleck
```

Outro exemplo de utilização, em que se mantém apenas as faces da Michelle Obama, configura-se duas camadas (tamanhos 1024 e 512 neurônios), e treina-se por 3 épocas:

```
python anonimify.py oscar.mp4 -pi Ben_Affleck Michelle_Obama
-l 1024 512 -e 3
```

4.5 Manipulação do reconhecimento das faces

Optou-se por avaliar a acurácia da anonimização através da comparação do resultado obtido em 4.4.3 com a rotulação manual. Para facilitar essa rotulação, desenvolveu-se um *software* que é capaz de ler o arquivo gerado após o processamento do vídeo, e mostrar ao usuário - quadro a quadro - o reconhecimento das faces. A partir disso, o usuário pode alterar o rótulo da pessoa reconhecida e seu grau de confiança, e por fim, salvar o arquivo.

Posteriormente, na etapa de testes, este *software* será utilizado para comparar o arquivo gerado pelo modelo treinado com o arquivo corrigido.

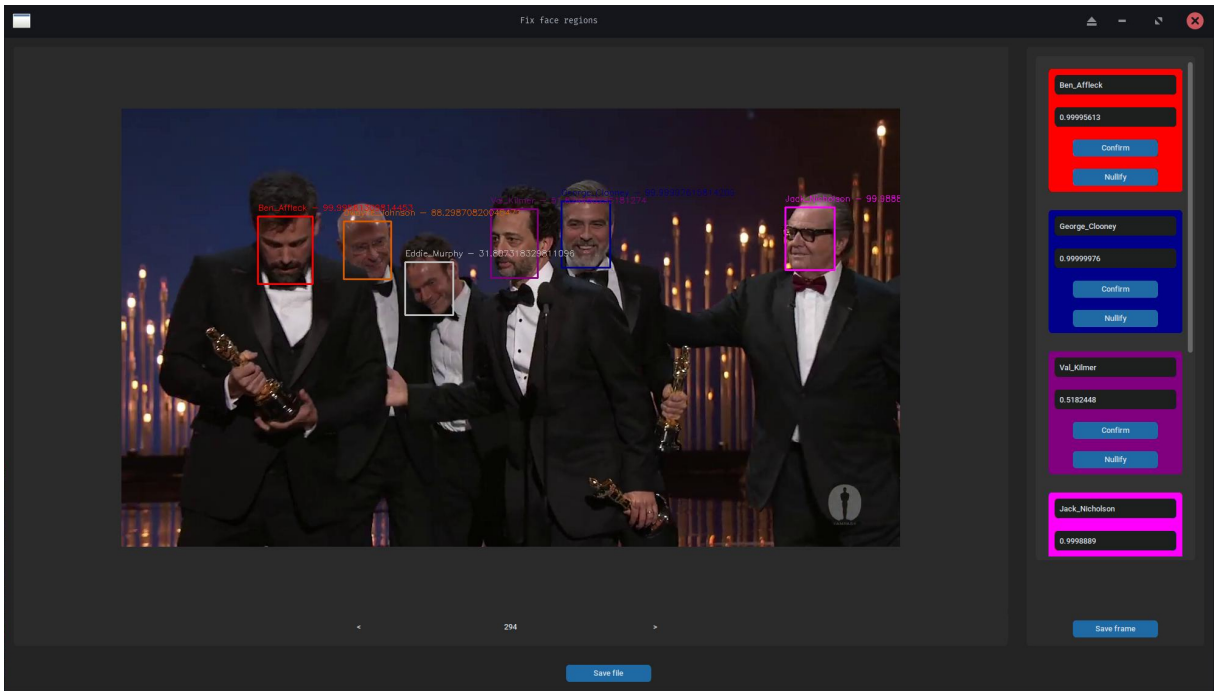


Figura 7 – *Software* para manipulação do reconhecimento das faces

5 Testes

Ao todo, 672 modelos de MLP foram treinados, utilizando mais de 15 mil fotos, contendo mais 160 identidades diferentes, com a finalidade de criar a ferramenta mais eficaz possível. Este capítulo discorrerá desses testes realizados.

5.1 Infraestrutura

Todos os testes foram realizados em um computador com distribuição Linux Manjaro 22.0.0, processador AMD Ryzen 5 3600X de 6 *cores* a 3,8 GHz, 16GB de RAM DDR4, e uma placa de vídeo NVIDIA GeForce RTX 2060 SUPER com 8GB de VRAM.

5.2 Conjunto de dados

5.2.1 Vídeo alvo

Os testes foram realizados em um um vídeo[1] encontrado na plataforma YouTube, que consiste em um trecho da premiação de cinema, em que o filme Argo recebe o prêmio de melhor filme, em 2013. O vídeo foi cortado com a finalidade de realizar diversos testes em um período de tempo menor (dez segundos; versão completa possui quase oito minutos).

Optou-se por esse vídeo, pois possui apenas pessoas públicas, das quais imagens são facilmente encontradas na *internet*, facilitando também a geração de um *dataset* que será utilizado para treinar o MLP.

Para todos os testes realizados nesse vídeo, utilizaram-se as seguintes pessoas a terem suas identidades mantidas no vídeo: Ben Affleck, George Clooney, Jack Nicholson e Michelle Obama. Essas quatro pessoas serão denominadas de *Preserved Identities* (PI, em português Identidades Preservadas). Do mesmo modo, as outras pessoas que farão parte do treino da rede neural, mas que não terão suas identidades mantidas em vídeo, serão chamadas por *Hidden Identities* (HI, em português Identidades Ocultas).

5.2.2 *Dataset* de treino para o MLP

Para a criação de um *dataset* foi criado um *script* em Python que utiliza a biblioteca *icrawler*[2] para baixar imagens de uma lista atores e pessoas públicas. Foi criada uma lista com

[1] <https://youtu.be/FtLKn5Y1ulc>
[2] <https://icrawler.readthedocs.io/>

163 nomes, entre eles atores, atrizes e políticos norte americanos. Para cada uma dessas pessoas foram baixadas cerca de cem imagens.

5.3 Estrutura do MLP

Foi desenvolvido um *script*, também em Python, para a preparar o treino dos mais diversos MLPs, contendo quantidades de camadas, números de neurônios por camada oculta, e épocas de treino distintas.

5.4 Limiar de confiança do reconhecimento

Conforme descrito em 4.3.2 e 4.3.3, é necessário configurar um limiar de confiança. Todos os testes foram realizados com limiar configurado em 0,8, que pode ser interpretado como "80% de confiança no reconhecimento".

5.5 Avaliação da acurácia

Primeiramente, gerou-se com o *software* descrito em 4.5 um arquivo de reconhecimento manual do vídeo alvo. A comparação segue o seguinte algoritmo:

Caso seja uma face que deveria ser **mantida**: acertou o nome (a etiqueta) e o grau de confiança é maior que o configurado: +1 acerto. Caso contrário: +1 erro.

Caso seja uma face que deveria ser **anonimizada**: reconheceu sendo uma PI (etiquetou com um dos nomes cuja face deveria ser mantida), e o grau de confiança é maior que o configurado: +1 erro. Caso contrário: +1 acerto.

Dessa forma, pontua quando há o correto reconhecimento, e não pontua caso a ferramenta mostre uma face que deveria ser anonimizada.

Enfim, a acurácia é calculada com:

$$acerto/(acerto + erro)$$

5.6 Resultados

5.6.1 Experimento

Foram treinadas todas as redes neurais que acomodassem todas as combinações de das seguintes características:

- número de camadas: de 1 a 3;

- número de neurônios por camada escondida: 256, 512, 1024 e 2048;
- épocas de treino: 2 e 3;
- quantidade de HI: 4, 8, 64, 128.

Dessa forma, a quantidade de testes realizados foi de 672, que demorou quase 13 horas de execução, e pode-se observar alguns comportamentos, que serão descritos abaixo.

5.6.2 Análise do impacto das características

Para encontrar se há variáveis que impactam mais ou menos a acurácia do modelo, esmiuçar-se-ão os diferentes modelos gerados, a partir de cada umas das características descritas.

5.6.2.1 Épocas de treino

Pode-se analisar visualmente na Figura 8 que, ao isolar apenas a quantidade de épocas de treino, essa variável parece não ser tão relevante no impacto na acurácia.

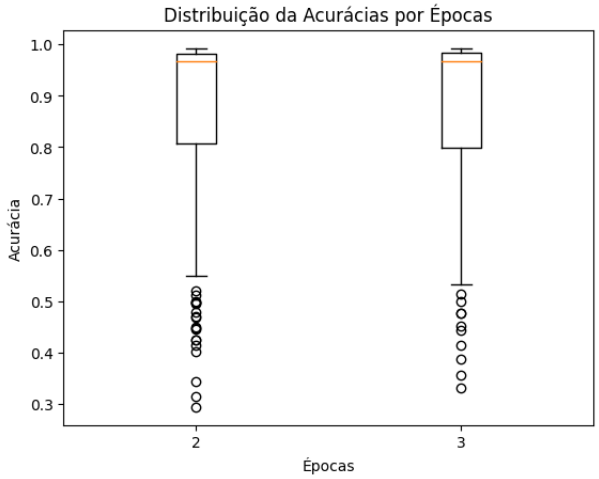


Figura 8 – Diagrama de caixa relativo às épocas de treino

5.6.2.2 Quantidade de camadas

Diferentemente das épocas, a quantidade de camadas parece ser diretamente relacionada à acurácia, como pode ser analisado na Figura 9. Apesar das medianas estarem próximas, o limite inferior da caixa de apenas uma camada é maior, o que indica que a maior parte das redes contendo apenas uma camada possui, em geral, uma acurácia maior.

Devido à maneira que os testes foram elaborados, há mais testes com 3 camadas do que com 2 e 1, e mais testes com 2 camadas do que com 1, pode-se também analisar a proporção da

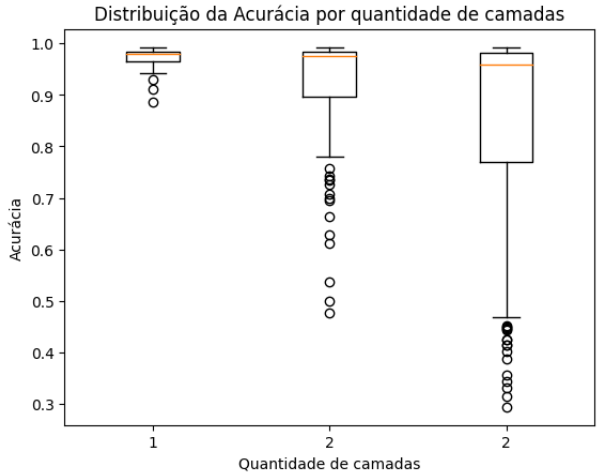


Figura 9 – Diagrama de caixa relativo à quantidade de camadas

Número de camadas	Quantidade total	Quantidade no top 100	Proporção
1	32	8	25,00%
2	128	24	18,75%
3	512	68	13,28%

Tabela 1 – Proporção das ocorrências de cada número de camadas no top 100

ocorrências de cada quantidade no top 100 testes, ordenados pela acurácia. Dessa forma, tem-se o seguinte resultado:

Conforme avaliado na Tabela 1, a proporção de testes com apenas uma cada no top 100 é maior quando comparado com as demais quantidades.

5.6.2.3 Quantidade de neurônios

Dessa forma, pode-se avaliar o impacto da quantidade de neurônios quando há apenas uma única camada. A distribuição da acurácia, para cada caso, pode ser visualizado na Figura 10, onde é notável que os testes realizados com 1024 neurônios tendem a performar melhor.

5.6.2.4 Quantidade de pessoas conhecidas

Uma das hipóteses levantadas durante a construção deste trabalho, é a de que quanto mais pessoas a MLP souber reconhecer, mais eficaz será. Ou seja, quando se detectar um rosto conhecido, que a probabilidade de acerto seja alta. Por outro lado, quando detectar um rosto que não conhece, apesar de reconhecer como alguma das pessoas contidas no *dataset* de treino (como é a natureza de um MLP), a probabilidade seja o mais baixa possível, de preferência abaixo do *threshold* configurado.

Dessa forma, ao analisar a distribuição da acurácia para cada quantidade de HI (Figura 11), além das quatro PI, pode-se notar que há uma melhora significativa devida ao aumento de identidades no treino.

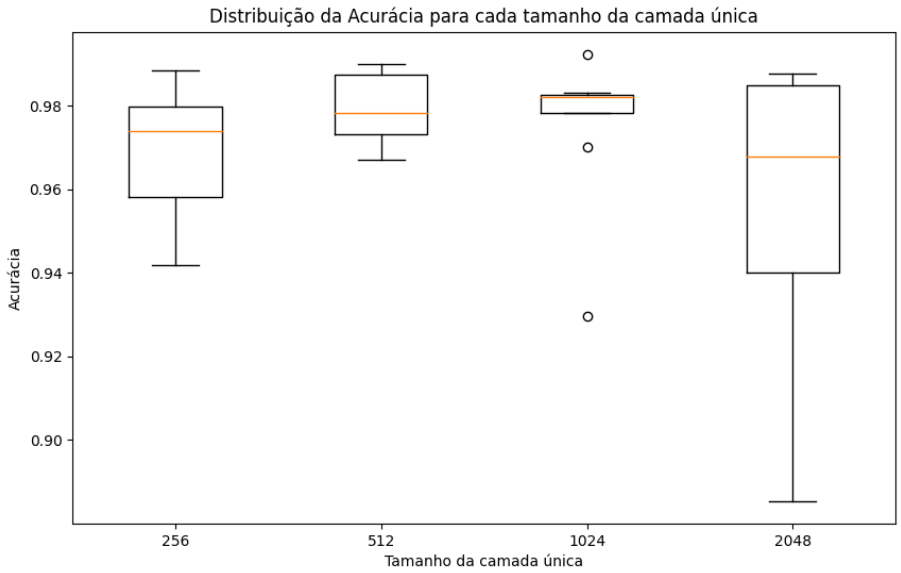


Figura 10 – Diagrama de caixa relativo à quantidade de neurônios quando há uma única camada

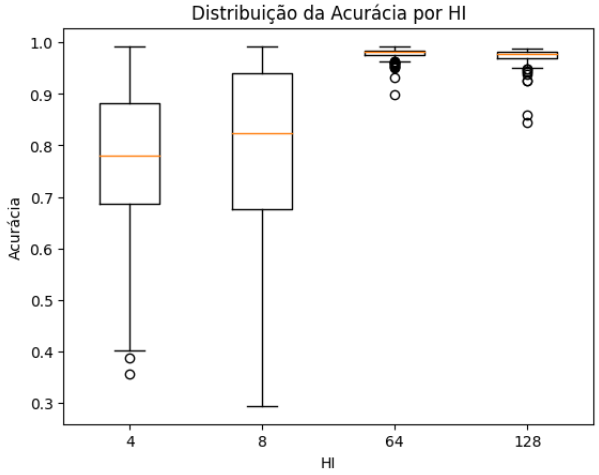


Figura 11 – Diagrama de caixa relativo à quantidade de HI

5.7 Análise dos Resultados

Todos os modelos com acurácia acima de 99%, podem ser encontrados na Tabela 2.

A mediana da acurácia de todos os modelos treinados é de aproximadamente 96,7%, ou seja, a grande maioria dos modelos performou muito bem. Além disso, o melhor modelo (chamado de "1024_3_4PI_8HI") foi capaz de acertar 1299 reconhecimentos, de um total de 1309 faces detectadas no vídeo testado, correspondendo a uma acurácia de 99,24%. Na Figura 12 encontram-se dois exemplos onde as quatro faces desejadas foram corretamente reconhecidas.

Na Figura 13, encontram-se dois quadros do vídeo alvo após o processo de anonimização. É possível ver que ambos Ben Affleck e George Clooney foram reconhecidos no quadro 217. Porém, à medida que o George Clooney se movimenta para trás do Jack Nicholson, seu rosto fica obstruído, e a rede se confunde, não mais o reconhecendo.

Camada 1	Camada 2	Camada 3	Épocas	Quantidade de HI	Acurácia
1024			3	8	99,24%
256	256		2	4	99,16%
2048	2048	512	3	64	99,16%
256	256		2	8	99,08%
256	1024	256	3	64	99,01%
256	1024		2	8	99,01%
2048	256		3	64	99,01%
2048	512	512	3	64	99,01%
2048	512		3	64	99,01%
256	512	1024	2	8	99,01%
256	256	256	3	8	99,01%
512			2	8	99,01%

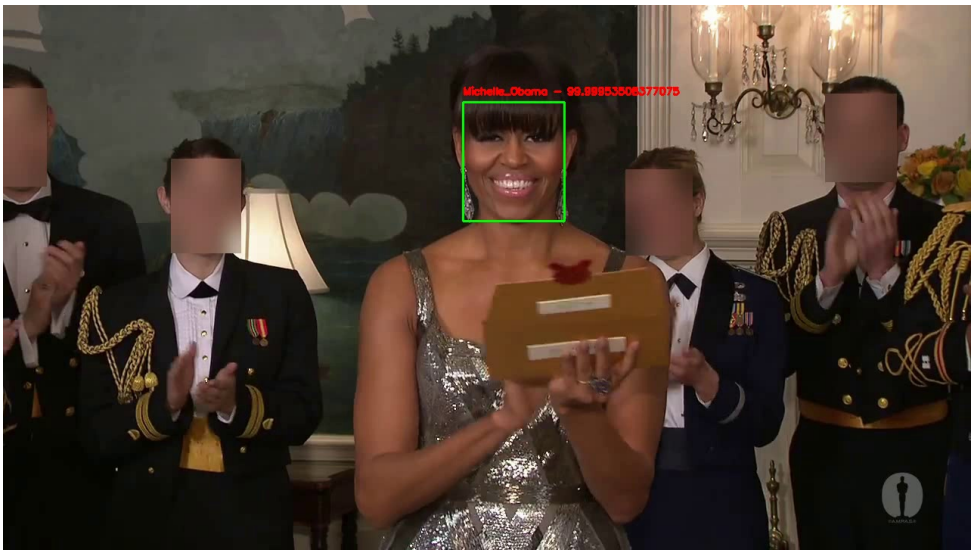
Tabela 2 – Proporção das ocorrências de cada número de camadas no top 100

Um ser humano é capaz de compreender o contexto do movimento que o George Clooney está realizando. Dessa forma, mesmo que a face detectada seja a da Figura 14 (bastante obstruída), somos capazes de reconhecer como sendo a mesma pessoa.

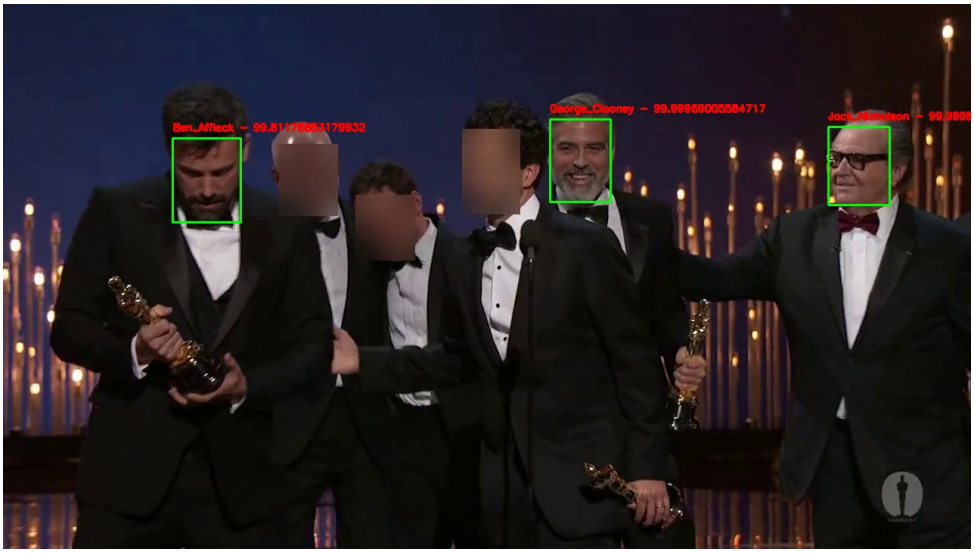
Vale destacar que, nesse caso específico, a ferramenta optou por anonimizar essa face não por a reconhecer erroneamente, mas sim por falta de confiança. Mais profundamente, no quadro anterior (217), a rede reconheceu sendo o George Clooney com uma probabilidade acima de 99%, enquanto no quadro seguinte (Figura 14) também a reconheceu sendo o George Clooney, porém com a probabilidade de apenas 67% (abaixo do limiar configurado), portanto a anonimizando.

Outro destaque importante é que todos os dez erros foram por anonimizar uma face que deveria ser mantida, e não por mostrar uma face que deveria ser anonimizada. Dessa forma, a ferramenta não deixou de cumprir o seu propósito.

Ainda assim, há bastante margem para mais testes, que podem incluir outros vídeos, de naturezas distintas, e o aumento de combinações de variáveis, como por exemplo mais tempo de treino (número maior de épocas). Realizar experimentos com outros vídeos serviria para confirmar que os resultados encontrados para um vídeo, replicam-se para outros. Não podemos afirmar que o melhor modelo encontrado nos experimentos deste trabalho será também o melhor modelo para outro vídeo.



(a) Michelle Obama reconhecida



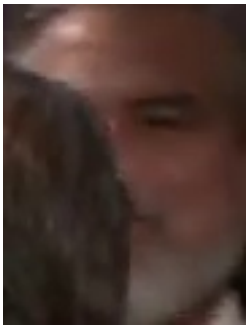
(b) Ben Affleck, Jack Nicholson e George Clooney reconhecidos

Figura 12 – Demonstração do correto reconhecimento das pessoas desejadas

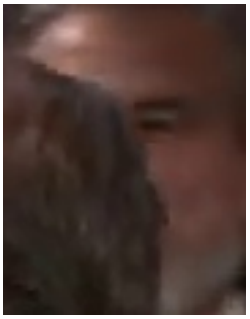
(a) Quadro 217/300

(b) Quadro 218/300

Figura 13 – Comparação dos quadros onde há falha do reconhecimento



(a) Face do George Clooney (quadro 217)



(b) Face do George Clooney (quadro 218)

Figura 14 – Comparação das faces onde há falha do reconhecimento

6 Conclusão

6.1 Considerações Finais

Dados os resultados apresentados no capítulo anterior, concluiu-se que a ferramenta desenvolvida atingiu satisfatoriamente o resultado esperado. Que no melhor dos casos anonimizou todas as faces que deveria, necessitando de pouquíssimo trabalho de ajuste para atingir a perfeição. Foi possível perceber que algumas configurações de MLP tendem a gerar redes mais precisas, porém com a mediana da acurácia em 96,7%, pode-se afirmar que essa técnica é bem flexível e pode ser utilizada para a finalidade deste trabalho. Para afirmações mais específicas sobre as diferentes configurações de MLP, mais testes são necessários, como, por exemplo, verificar se a acurácia de uma rede em um vídeo se mantém para outro vídeo. Espera-se que este trabalho sirva de inspiração para que demais trabalhos sejam desenvolvidos nessa área, uma vez que a privacidade é um bem muito importante, e provamos que é possível desenvolver uma automação eficaz, reduzindo consideravelmente o esforço manual humano.

6.2 Trabalhos Futuros

Durante todo o processo de desenvolvimento deste trabalho, surgiram diversas ideias para trabalhos futuros. Ideias essas que envolvem ampliar a aplicabilidade deste trabalho, melhorar sua performance, ou facilitar seu uso. Alguns possíveis trabalhos futuros encontram-se listados abaixo.

6.2.1 Interface gráfica

Para facilitar o uso de pessoas com nenhum conhecimento em programação, seria de bastante importância a implementação de uma interface gráfica amigável e fácil de usar, que permitisse aos usuários mais avançados a possibilidade de ajustar parâmetros.

Outra funcionalidade possível é a edição das regiões de faces detectadas e reconhecidas. Essa edição permitiria o usuário facilmente remover falsos negativos/positivos. Ressalta-se que essas regiões já são salvas em um arquivo separadamente, o que facilitaria a implementação dessa funcionalidade.

6.2.2 Análise de contexto

Ao processar cada *frame* individualmente, ocasiona-se um dos principais motivos de falha deste trabalho, que é a falta de entendimento de contexto entre *frames*, isto é, saber que a pessoa detectada no *frame* atual é a mesma do *frame* anterior e seguinte. Este é um problema,

pois, tratando-se de vídeos, podem ocorrer embaçamento da face devido ao movimento da pessoa ou câmera (*motion blur*), ou até mesmo a pessoa mover a cabeça em um ângulo em que seja virtualmente impossível reconhecê-la, mesmo para um humano familiar com aquela pessoa.

Um possível trabalho futuro seria adicionar a questão contexto ao *pipeline* deste trabalho.

6.2.3 Anonimização com ofuscação de atributos

Conforme citado anteriormente neste trabalho, a ofuscação de atributos é bastante interessante para manter a sutileza das anonimizações realizadas no vídeo. Para tal, é necessário desenvolver, ou analisar e avaliar as soluções já existentes, e adaptar este trabalho para ofuscar os atributos, no lugar do filtro de embaçamento.

6.2.4 Anonimização de crianças e bebês

Pessoas abaixo da maioridade penal podem ser alvos para um trabalho de anonimização automática de pessoas. Seria interessante um trabalho que generalizasse este projeto, inserindo a possibilidade de anonimizar pessoas baseadas em suas idades. A grande dificuldade, aos olhos do autor deste trabalho, está em evitar falso positivos ou negativos, uma vez que existe um alto grau de sutileza em relação à aparência e idade real.

6.2.5 Avaliar alternativas à MLP

Um MLP não é a única maneira possível de classificar pessoas utilizando os *embeddings* gerados. Um outro método menos inteligente, por assim dizer, é utilizar similaridade por cosseno[1], que consiste em comparar dois vetores utilizando a função trigonométrica cosseno. Por outro lado, também existem outras formas de utilizar MLPs, como proposto por Capello et al. 2009, que propõe a utilização de, ao invés de um único MLP com múltiplas classes, um MLP por classe.

[1] <https://pt.wikipedia.org/wiki/Similaridade_por_cosseno>

Referências

BARBU, T. Gabor filter-based face recognition technique. *Proceedings of the Romanian Academy*, v. 11, n. 3, p. 277–283, 2010. Citado na página 21.

BOTALB, A. et al. Contrasting convolutional neural network (cnn) with multi-layer perceptron (mlp) for big data analysis. In: IEEE. *2018 International conference on intelligent and advanced system (ICIAS)*. [S.l.], 2018. p. 1–5. Citado na página 27.

CANDIDO, J. P. S.; ARAÚJO, T. F. de; RIBEIRO, W. A. C. *Histórico da Lei Geral de Proteção de Dados (LGPD)*. 2021. Disponível em: <https://advocatta.org/historico-da-lei-geral-de-protecao-de-dados-lgpd/>. Citado na página 21.

Cao, Z. et al. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. Citado na página 36.

CAPELLO, D. et al. Array of multilayer perceptrons with no-class resampling training for face recognition. AEPIA, 2009. Citado na página 54.

DENG, J. et al. Retinaface: Single-shot multi-level face localisation in the wild. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 5203–5212. Citado na página 21.

DENG, J. et al. Arcface: Additive angular margin loss for deep face recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2019. Citado na página 30.

DESAI, M.; SHAH, M. An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (mlp) and convolutional neural network (cnn). *Clinical eHealth*, Elsevier, v. 4, p. 1–11, 2021. Citado na página 27.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998. Citado na página 27.

GROSSELFINGER, A.-K.; MÜNCH, D.; ARENS, M. An architecture for automatic multimodal video data anonymization to ensure data protection. In: BOUMA, H. et al. (Ed.). *Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies III*. SPIE, 2019. v. 11166, p. 206 – 217. Disponível em: <https://doi.org/10.1117/12.2533031>. Citado 2 vezes nas páginas 21 e 36.

JADHAV, A. et al. Survey on face detection algorithms. *International Journal of Innovative Science and Research Technology*, v. 6, n. 2, 2021. Citado 2 vezes nas páginas 21 e 28.

KORTLI, Y. et al. Face recognition systems: A survey. *Sensors*, MDPI, v. 20, n. 2, p. 342, 2020. Citado 2 vezes nas páginas 13 e 29.

KUMAR, A.; KAUR, A.; KUMAR, M. Face detection techniques: a review. *Artificial Intelligence Review*, Springer, v. 52, n. 2, p. 927–948, 2019. Citado 3 vezes nas páginas 13, 27 e 28.

LI, J. et al. Identity-preserving face anonymization via adaptively facial attributes obfuscation. In: *Proceedings of the 29th ACM International Conference on Multimedia*. [S.l.: s.n.], 2021. p. 3891–3899. Citado na página 31.

LIU, W. et al. Sphereface: Deep hypersphere embedding for face recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 212–220. Citado na página 30.

PULFER, E.-M. Different approaches to blurring digital images and their effect on facial detection. 2019. Citado na página 30.

QI, D. et al. Yolo5face: why reinventing a face detector. *arXiv preprint arXiv:2105.12931*, 2021. Citado na página 29.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado na página 29.

SHAMS, B.; NASIM, F. I. et al. *Analysis on face recognition based on five different viewpoint of face images using MTCNN and FaceNet*. Tese (Doutorado) — Brac University, 2019. Citado na página 21.

SHARIF, M. et al. Face recognition: A survey. *Journal of Engineering Science & Technology Review*, v. 10, n. 2, 2017. Citado na página 21.

SMELYAKOV, K. et al. The neural network models effectiveness for face detection and face recognition. In: *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*. [S.l.: s.n.], 2021. p. 1–7. Citado na página 21.

TRAN, P. Fast video-based face recognition in collaborative learning environments. *arXiv preprint arXiv:2110.14720*, 2021. Citado 4 vezes nas páginas 28, 29, 30 e 33.

TURK, M. Eigenfaces and beyond. *Face Processing: Advanced Modeling and Methods*, Citeseer, p. 55–86, 2005. Citado na página 21.

YANG, S. et al. Wider face: A face detection benchmark. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado na página 27.

ZHOU, J.; PUN, C.-M. Personal privacy protection via irrelevant faces tracking and pixelation in video live streaming. *IEEE Transactions on Information Forensics and Security*, IEEE, v. 16, p. 1088–1103, 2020. Citado 2 vezes nas páginas 33 e 34.

Apêndices

APÊNDICE A – Código-fonte

Arquivo: anonimify/requirements.txt

```
absl-py==1.4.0
albumentations==1.3.0
astunparse==1.6.3
cachetools==5.3.1
certifi==2023.5.7
charset-normalizer==3.1.0
coloredlogs==15.0.1
conda==23.1.0
conda-package-handling==1.8.1
contourpy==1.0.7
customtkinter==5.1.3
cycler==0.11.0
Cython==0.29.35
darkdetect==0.8.0
easydict==1.10
flatbuffers==23.5.26
fonttools==4.39.4
gast==0.4.0
google-auth==2.19.1
google-auth-oauthlib==1.0.0
google-pasta==0.2.0
grpcio==1.54.2
h5py==3.8.0
humanfriendly==10.0
idna==3.4
imageio==2.31.0
insightface==0.7.3
jax==0.4.11
joblib==1.2.0
keras==2.12.0
kiwisolver==1.4.4
lazy_loader==0.2
libclang==16.0.0
```

```
Markdown==3.4.3
MarkupSafe==2.1.3
matplotlib==3.7.1
ml-dtypes==0.2.0
mpmath==1.3.0
networkx==3.1
numpy==1.23.5
nvidia-cublas-cu11==11.11.3.6
nvidia-cudnn-cu11==8.6.0.163
oauthlib==3.2.2
onnx==1.14.0
onnxruntime-gpu==1.15.0
opencv-python==4.7.0.72
opencv-python-headless==4.7.0.72
opt-einsum==3.3.0
packaging==23.1
Pillow==9.5.0
pluggy==1.0.0
prettytable==3.7.0
protobuf==4.23.2
pyasn1==0.5.0
pyasn1-modules==0.3.0
pycosat==0.6.3
pyparsing==3.0.9
python-dateutil==2.8.2
python-version==0.0.2
PyWavelets==1.4.1
PyYAML==6.0
qudida==0.0.4
requests==2.31.0
requests-oauthlib==1.3.1
rsa==4.9
ruamel.yaml==0.17.31
ruamel.yaml.clib==0.2.7
scikit-image==0.21.0
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
sympy==1.12
```

```
tensorboard==2.12.3
tensorboard-data-server==0.7.0
tensorflow==2.12.0
tensorflow-estimator==2.12.0
tensorflow-io-gcs-filesystem==0.32.0
termcolor==2.3.0
threadpoolctl==3.1.0
tifffile==2023.4.12
tqdm==4.65.0
typing_extensions==4.6.3
urllib3==1.26.16
wcwidth==0.2.6
Werkzeug==2.3.4
wrapt==1.14.1
```

Arquivo: anonimify/anonimify.py

```
import os
import sys
import argparse
import pickle
from src.KnownFaceEmbedder import KnownFaceEmbedder
from src.KnownEmbeddingTrainer import KnownEmbeddingTrainer
from src.VideoAnonymizer import VideoAnonymizer

def create_file_structure():
    os.makedirs("outputs", exist_ok=True)

def get_args():
    parser = argparse.ArgumentParser(prog="Anonimify",
        description="Anonimize people in video")
    parser.add_argument('filename')
    parser.add_argument('-pi', '--preserved-identities', nargs
        ='+', default=[], metavar="Preserved Identities")
    parser.add_argument('-l', '--layers', nargs='+', default
        =[256, 256, 512], type=int, metavar="Layers")
```

```
    parser.add_argument('-e', '--epochs', default=1, type=int,
        metavar="Epochs")
    parser.add_argument('-le', '--labeled-embeddings', metavar
        ="Labeled Embeddings file")

    return parser

def parse_args(parser):
    args = parser.parse_args()
    return args

def create_labeled_embeddings(labeled_embeddings_filename):
    embedder = KnownFaceEmbedder()

    if labeled_embeddings_filename == None:
        embedder.process_images("datasets/")
        embedder.save_file(path=f"outputs/labeled_embeddings.
            pickle")
        return embedder

    embedder.load_embeddings_from_file(
        labeled_embeddings_filename)

    return embedder

def train_known_labeled_embeddings(embedder, layers, epochs):
    trainer = KnownEmbeddingTrainer(embedder.labeled_embeddings
        (), layers=len(layers), units_per_layer=layers, epochs=
        epochs)
    trainer.train(model_name="trained_known_labeled_embeddings
        ")

def anonymize_video(video_filename, preserved_identities,
    embedder):
    anonymizer = VideoAnonymizer(embedder.labeled_embeddings(),
        "outputs/encoded_labels.pickle")
    regions = anonymizer.save_regions(video_filename, "outputs/
        trained_known_labeled_embeddings.h5", "outputs/
        face_regions.pickle")
```

```
    anonymizer.anonymize_regions(video_filename, f"anonymized.
        mp4", regions, preserved_identities)

def main():
    create_file_structure()
    parser = get_args()
    args = parse_args(parser)

    embedder = create_labeled_embeddings(args.
        labeled_embeddings)
    train_known_labeled_embeddings(embedder, args.layers, args.
        epochs)
    anonymize_video(args.filename, args.preserved_identities,
        embedder)

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

Arquivo: anonimify/src/KnownFaceEmbedder.py

```
import os
import cv2
import numpy as np
import pickle
import insightface
from insightface.app import FaceAnalysis

class KnownFaceEmbedder:

    def __init__(self, analyzer_model="buffalo_l"):
        self.analyzer = FaceAnalysis(name=analyzer_model,
            providers=['CUDAExecutionProvider', '
            CPUExecutionProvider'])
        self.labels = []
        self.embeddings = []
        self.analyzer.prepare(ctx_id=0)
```

```
def load_embeddings_from_file(self, filename):
    with open(filename, "rb") as f:
        labeled_embeddings = pickle.load(f)

    self.labels = labeled_embeddings["labels"]
    self.embeddings = labeled_embeddings["embeddings"]

    # necessary to perform some imread previously,
       otherwise it gives memory error
    img = cv2.imread("/home/los/Documents/ufsc/tcc/
       new_tests/datasets/Ben_Affleck/000001.jpg")
    self.analyzer.get(img)

def generate_embedding(self, image, label):
    img_results = self.analyzer.get(image)

    if len(img_results) == 1:
        self.labels.append(label)
        self.embeddings.append(img_results[0].embedding)

        return img_results[0]

    print(f"Multiple faces detected: {len(img_results)}")
    return None

def process_images(self, path = "datasets/images/"):
    for known_label in os.listdir(path):
        for filename in os.listdir(f"{path}/{known_label}")
           :
            img = cv2.imread(f"{path}/{known_label}/{
               filename}")
            img_data = self.generate_embedding(img,
               known_label)

def process_videos(self, path = "datasets/videos/",
   skip_frames=10, rotate=False):
    for known_label in os.listdir(path):
        for filename in os.listdir(f"{path}/{known_label}")
           :
```

```
                video = cv2.VideoCapture(f"{path}/{known_label
                    }/{filename}")
                count = 0

                while video.isOpened():
                    ret, frame = video.read()

                    if (rotate):
                        frame = cv2.rotate(frame, cv2.
                            ROTATE_180)

                    if ret:
                        self.generate_embedding(frame,
                            known_label)
                        img_data = self.generate_embedding(
                            frame, known_label)
                        count += skip_frames
                        video.set(cv2.CAP_PROP_POS_FRAMES,
                            count)


                    else:
                        video.release()
                        break

    def process(self):
        self.process_images()
        self.process_videos()

    def labeled_embeddings(self):
        return {"embeddings": self.embeddings, "labels": self.
            labels}

    def save_file(self, path = "outputs/processed/
        labeled_embeddings.pickle"):
        f = open(path, "wb")
        f.write(pickle.dumps(self.labeled_embeddings()))
        f.close()
```

Arquivo: anonimify/src/KnownEmbeddingTrainer.py

```
from src.models.SoftMax import SoftMax
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
import numpy as np
import pickle
import time
from tensorflow.keras.callbacks import TensorBoard

class KnownEmbeddingTrainer:

    def __init__(self, labeled_embeddings, layers=2,
        units_per_layer=4096, dropout=0.2, batch_size=32, epochs
        =5):
            self.labeled_embeddings = labeled_embeddings
            self.layers = layers
            self.units_per_layer = units_per_layer
            self.dropout = dropout
            self.batch_size = batch_size
            self.epochs = epochs
            self.label_encoder = None

    def __encoded_labels(self):
            self.label_encoder = LabelEncoder()
            labels = self.label_encoder.fit_transform(self.
                labeled_embeddings["labels"])
            num_classes = len(np.unique(labels))
            labels = labels.reshape(-1, 1)
            one_hot_encoder = OneHotEncoder()
            return (num_classes, one_hot_encoder.fit_transform(
                labels).toarray())

    def train(self, model_name=None, encoded_labels_path="
        outputs/encoded_labels.pickle"):
```

```
NAME = f"{self.layers}-dense-{self.units_per_layer}-
    nodes-{self.epochs}-epochs__{int(time.time())}"

num_classes, labels = self.__encoded_labels()
embeddings = np.array(self.labeled_embeddings["
    embeddings"])
input_shape = embeddings.shape[1]


model = SoftMax(input_shape=(input_shape,), num_classes
    =num_classes, layers=self.layers, units_per_layer=
    self.units_per_layer, dropout=self.dropout)

training_model = model.build()

cv = KFold(n_splits = 2, random_state = 42, shuffle=
    True)

tensorboard = TensorBoard(log_dir="logs/{}".format(
    model_name or NAME))

for train_idx, valid_idx in cv.split(embeddings):
    X_train, X_val, y_train, y_val = embeddings[
        train_idx], embeddings[valid_idx], labels[
        train_idx], labels[valid_idx]

    training_model.fit(X_train, y_train,
        batch_size=self.batch_size,
        epochs=self.epochs,
        verbose=1,
        validation_data=(X_val, y_val),
        callbacks=[tensorboard])

training_model.save(f"outputs/{model_name or NAME}.h5")
f = open(encoded_labels_path, "wb")
f.write(pickle.dumps(self.label_encoder))
f.close()
```

Arquivo: anonimify/src/VideoAnonymizer.py

```
from insightface.app import FaceAnalysis
from keras.models import load_model
import pickle
import cv2
import numpy as np
import time

class VideoAnonymizer:

    def __init__(self, labeled_embeddings, encoded_labels_path)
        :
        with open(encoded_labels_path, "rb") as f:
            self.encoded_labels = pickle.load(f)

        self.embeddings = np.array(labeled_embeddings['
            embeddings'])
        self.labels = self.encoded_labels.fit_transform(
            labeled_embeddings['labels'])

        model_pack_name = 'buffalo_l'
        self.face_embedder = FaceAnalysis(name=model_pack_name,
            providers=['CUDAExecutionProvider', '
            CPUExecutionProvider'])
        self.face_embedder.prepare(ctx_id=0)

    def anonymize_regions(self, video_path, anonymized_path,
        frames_regions, people_to_keep):
        start = time.time()
        video = cv2.VideoCapture(video_path)

        frame_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
        frame_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT)
            )
        total_frame_count = int(video.get(cv2.
            CAP_PROP_FRAME_COUNT))
        frame_count = 0
```

```
fps = int(video.get(cv2.CAP_PROP_FPS))
anonymized = cv2.VideoWriter(anonymized_path, cv2.
    VideoWriter_fourcc("m","p","4","v"), fps, (
    frame_width, frame_height))

while(video.isOpened()):
    ret, image = video.read()

    if ret == True:
        percentage = 100 * (frame_count + 1) /
            total_frame_count
        filled = int(percentage)
        empty = int(100-filled)

        end = time.time()
        hours, rem = divmod(end-start, 3600)
        minutes, seconds = divmod(rem, 60)
        pretty_time = "{:0>2}:{:0>2}:{:05.2f}".format(
            int(hours),int(minutes),seconds)
        print(f'\rElapsed time: {pretty_time} - [{"="*
            filled}{" "*empty}] {"{:.1f}".format(
            percentage)}% ', end='')

        for frame_region in frames_regions[frame_count
            ]:
            x, y = frame_region["box"][0][0],
                frame_region["box"][0][1]
            w, h = frame_region["box"][1][0] -
                frame_region["box"][0][0], frame_region
                ["box"][1][1] - frame_region["box
                "][0][1]

            roi = image[y:y+h, x:x+w]
            cv2.imwrite(f"outputs/faces/{frame_count}-{
                frame_region['label']}.jpg", roi)

            if not(frame_region["label"] in
                people_to_keep) or frame_region["
                probability"] < 0.8:
```

```
                        x, y = frame_region["box"][0][0],
                            frame_region["box"][0][1]
                        w, h = frame_region["box"][1][0] -
                            frame_region["box"][0][0],
                            frame_region["box"][1][1] -
                            frame_region["box"][0][1]

                        roi = image[y:y+h, x:x+w]
                        blur = cv2.GaussianBlur(roi, (101, 101)
                            , 0)

                        image[y:y+h, x:x+w] = blur
                    else:
                        text = "{} - {}".format(frame_region["
                            label"], frame_region["probability
                            "]*100)

                        y = frame_region["box"][0][1] - 10 if
                            frame_region["box"][0][1] - 10 > 10
                            else frame_region["box"][0][1] + 10
                        cv2.putText(image, text, (int(
                            frame_region["box"][0][0]), int(y)),
                             cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0,
                             0, 255), 2)
                        cv2.rectangle(image, frame_region["box
                            "][0], frame_region["box"][1],
                            (0,255,0), 2)

                frame_count += 1
                anonymized.write(image)
            else:
                break

        print()
        video.release()
        anonymized.release()

    def save_regions(self, video_path, model_path, save_path):
        start = time.time()
```

```
video = cv2.VideoCapture(video_path)
model = load_model(model_path)
frame_count = 0
total_frame_count = int(video.get(cv2.
   CAP_PROP_FRAME_COUNT))

regions = []

while(video.isOpened()):
    ret, image = video.read()

    if ret == True:
        frame_count += 1

        percentage = 100 * frame_count /
           total_frame_count
        filled = int(percentage)
        empty = int(100-filled)
        end = time.time()
        hours, rem = divmod(end-start, 3600)
        minutes, seconds = divmod(rem, 60)
        pretty_time = "{:0>2}:{:0>2}:{:05.2f}".format(
           int(hours),int(minutes),seconds)
        print(f'\rElapsed time: {pretty_time} - [{"="*
           filled}{" "*empty}] {"{:.1f}".format(
           percentage)}% ', end='')

        img_results = self.face_embedder.get(image)

        if len(img_results) != 0:
            frame_regions = []

            for result in img_results:
                bbox = result.bbox
                embedding = result.embedding.reshape
                   (1,-1)

                preds = model.predict(embedding,
                   verbose=0)
```

```
                        preds = preds.flatten()

                        j = np.argmax(preds)
                        proba = preds[j]

                        top_left = (max(0, int(bbox[0])), max
                            (0, int(bbox[1])))
                        bottom_right = (max(0, int(bbox[2])),
                            max(0, int(bbox[3])))

                        name = self.encoded_labels.classes_[j]

                        frame_regions.append({"label": name, "
                            probability": proba, "box": [
                            top_left, bottom_right]})

                    regions.append(frame_regions)
                else:
                    regions.append([])


            else:
                break

        print()
        video.release()
        f = open(save_path, "wb")
        f.write(pickle.dumps(regions))
        f.close()

        return regions
```

Arquivo: anonimify/src/models/SoftMax.py

```
from keras.layers import Input, Dense, Dropout
from keras.models import Sequential
from keras.optimizers import Adam
```

```
import keras

class SoftMax():
    def __init__(self, input_shape, num_classes, layers,
       units_per_layer, dropout):
        self.input_shape = input_shape
        self.num_classes = num_classes
        self.layers = layers
        self.units_per_layer = units_per_layer
        self.dropout = dropout

    def build(self):
        print(f"{self.input_shape} {self.num_classes}")

        model = Sequential()
        model.add(Input(shape=self.input_shape))

        if type(self.units_per_layer) is list:
            for layer_index in range(len(self.units_per_layer))
               :
                model.add(Dense(self.units_per_layer[
                   layer_index], activation='relu'))
                model.add(Dropout(self.dropout))
        else:
            for _ in range(self.layers):
                model.add(Dense(self.units_per_layer,
                   activation='relu'))
                model.add(Dropout(self.dropout))


        model.add(Dense(self.num_classes, activation='softmax')
           )

        optimizer = Adam()
        model.compile(loss=keras.losses.
           categorical_crossentropy,
                      optimizer=optimizer,
                      metrics=['accuracy'])
        return model
```

APÊNDICE B – Artigo SBC

Detecção e reconhecimento de faces para a anonimização de pessoas em vídeos

Leonardo de Oliveira da Silva[1], **Alexandre Gonçalves Silva**[1]

[1]Universidade Federal de Santa Catarina

Abstract. *Several applications require the anonymization of individuals in videos, either for legal reasons, lack of consent for image usage, or even underage individuals. Naturally, the task of detecting and recognizing these individuals needs to be performed at some stage of the process. If it is done during video editing, post-recording, it tends to be costly and time-consuming, as it requires manual labor of a human cutting out the faces frame by frame. With recent technological advancements, it is now possible to leverage artificial intelligence to perform all stages of this process: detecting, recognizing, and anonymizing faces in videos. This innovation brings many advantages, such as eliminating the need for human labor and increasing the speed of the task, while also allowing anonymization to be done directly during recording, eliminating the need for video post-processing. In this context, this work utilizes existing face detection and recognition techniques to apply a filter that anonymizes unwanted faces and prevents their recognition. To achieve this, the techniques commonly used for this purpose were reviewed, and the ones that best suited the context of this work were selected. A tool was developed to perform automatic anonymization. Finally, its performance was analyzed using an accuracy metric by comparing the results of manual labeling with automated labeling. In total, 672 tests were conducted, and a median accuracy of 96.7% was obtained.*

Resumo. *Diversas aplicações requerem a anonimização de pessoas presentes em vídeos, seja por questões legais, falta de consentimento do uso de imagem, ou até pessoas menores de idade. Naturalmente, o trabalho de detectar e reconhecer essas pessoas deve ser feito em alguma etapa do processo, e caso seja feito na edição do vídeo, pós gravação, tende a ser custoso e demorado, pois exige o trabalho braçal de um ser humano cortando, quadro a quadro, os rostos encontrados. Com avanços tecnológicos recentes, pode-se aproveitar da inteligência artificial para realizar todas as etapas desse processo: detectar, reconhecer e anonimizar as faces presentes nos vídeos. Essa inovação traz consigo muitas vantagens, como a desalocação de um ser humano e o aumento da velocidade na realização do trabalho, além de permitir que a anonimização seja feita diretamente na gravação, de forma a retirar a necessidade de pós-processamento do vídeo. Nesse contexto, este trabalho utiliza técnicas de detecção e reconhecimento de faces já existentes para aplicar um filtro que anonimize os rostos indesejados e impossibilite o reconhecimento dos mesmos. Para isto, foram revisadas as técnicas normalmente utilizadas neste fim, e foram selecionadas as que melhor se aplicam no contexto deste trabalho, e desenvolveu-se uma ferramenta para realizar a anonimização automática. Por fim, analisou-se sua performance utilizando métrica de acurácia, comparando*

o resultado de uma rotulação manual com a rotulação automatizada. Ao todo, foram realizados 672 testes, e obteve-se mediana da acurácia de 96,7%.

1. Introdução

A privacidade dos dados pessoais torna-se, cada vez mais, importante no senso comum, e a possibilidade de identificação facial em vídeos não fica de fora. No Brasil, discute-se o tema com mais força desde 2010, quando foi realizada uma importante consulta pública. Desde então, diversas leis surgiram: em 2014 iniciou-se o Marco Civil da Internet, que intensificou na internet o direito à privacidade, e em 2020 entrou em vigor a LGPD (Lei Geral de Proteção de Dados) [Candido et al. 2021].

Ao realizar gravações de vídeo em público, é natural que pessoas indesejadas apareçam. Dessa forma, solicitar a permissão do uso de imagem de cada uma dessas pessoas pode não ser factível, dependendo do contexto da gravação. Portanto, faz-se necessário o uso de alguma edição das gravações para cortar o aparecimento dessas pessoas, ou anonimizá-las. O trabalho manual dessa anonimização usando *softwares* específicos para edição de vídeo tende a ser custoso e demorado, e em trabalhos como o de [Grosselfinger et al. 2019], conclui-se que pode levar até dez vezes mais tempo do que sua arquitetura automatizada proposta, utilizando IA (Inteligência Artificial).

Durantes as últimas duas décadas surgiram diversos algoritmos baseados em IA para a detecção de faces. As vantagens e desvantagens destes algoritmos normalmente estão relacionadas à precisão e tempo de processamento necessário para sua execução. Dessa forma, as necessidades da aplicação determinam qual algoritmo se adequa mais. Uma transmissão ao vivo requer rápida execução, por exemplo. Dentre os algoritmos pesquisados por [Smelyakov et al. 2021], RetinaFace[1] foi o algoritmo com a maior precisão da detecção, porém seu tempo de execução foi um dos maiores. Já na pesquisa de [Jadhav et al. 2021], que incluiu outros algoritmos, conclui-se que o algoritmo com maior precisão foi o MTCNN[2].

Detectar faces, dependendo da necessidade de cada aplicação, pode não ser o suficiente. Diversos algoritmos buscam reconhecer cada indivíduo, e assim conseguir diferenciar uma pessoa da outra. [Sharif et al. 2017] explora diversas das mais comuns técnicas na área de reconhecimento facial, sendo duas delas Eigenface[3], Gabor Wavelet[4].

Há diversas situações e contextos em que seja necessário anonimizar certas pessoas em um vídeo. Como dito anteriormente, filmagens em locais públicos, filmagens que apareçam pessoas abaixo da maioridade penal, ou até mesmo pessoas que explicitamente não deram permissão para o uso de imagem, são alguns dos exemplos da aplicação de anonimização de faces das quais se anonimiza apenas algumas das faces detectadas, não todas.

Este trabalho tem como objetivo desenvolver uma aplicação capaz de detectar e reconhecer faces para anonimizar apenas as pessoas selecionadas, utilizando técnicas precisas, mas não necessariamente rápidas, uma vez que a aplicação não visa sua utilização

[1] [Deng et al. 2020]
[2] [Shams et al. 2019]
[3] [Turk 2005]
[4] [Barbu 2010]

em vídeos de transmissão ao vivo. Portanto, o tempo de processamento - ainda que importante - não será uma métrica crítica.

1.1. Objetivos

Esta seção apresenta o objetivo geral e objetivos específicos deste trabalho.

1.1.1. Objetivo Geral

Desenvolver um programa que possibilite, através de técnicas de inteligência artificial para detecção e reconhecimento de faces, anonimizar as faces das pessoas selecionadas em um arquivo de vídeo.

1.1.2. Objetivos Específicos

- Analisar as principais técnicas de identificação de faces e reconhecimento de par de faces;
- Desenvolver um algoritmo para a anonimização das faces selecionadas;
- Avaliar a acurácia em comparação com a rotulação manual;
- Tornar público o código-fonte para uso da comunidade interessada.

1.2. Metodologia

Para o desenvolvimento desse trabalho, serão realizadas as etapas descritas a seguir:

1. revisão bibliográfica sistemática em relação às técnicas mais utilizadas e precisas para detecção e anonimização de faces;
2. desenvolvimento de um algoritmo que utilize as técnicas escolhidas;
3. desenvolvimento de um algoritmo que aplique algum filtro nas faces para anonimizá-las;
4. analisar os resultados utilizando a acurácia calculada;
5. redigir as documentações e relatórios necessários para as disciplinas de Introdução ao TCC, TCC I e TCC 2.

2. Fundamentação Teórica

Este capítulo apresenta os conceitos necessários para o entendimento e desenvolvimento do trabalho. São apresentados alguns dos métodos mais encontrados em trabalhos relacionados à detecção, reconhecimento e anonimização de faces.

2.1. Redes Neurais

2.1.1. Perceptron Multicamadas

Segundo [Gardner and Dorling 1998], o Perceptron Multicamadas (em inglês *Multilayer Perceptron*, MLP) é uma rede neural simples, com uma camada de entrada, camadas ocultas - que podem ser uma ou mais, e a camada de saída. MLP é totalmente conectada, significando que todos neurônios de uma camada são conectados na camada posterior. O perceptron multicamadas também tem como característica possuir uma direção de processamento de informações, portanto sendo considerado como uma rede neural *feed-forward*.

2.1.2. Redes Neurais Convolucionais

As Redes Neurais Convolucionais (em inglês *Convolutional Neural Network*, CNN) são bem parecidas com MLPs, em termos de estrutura. Também possui camadas de entrada, camadas ocultas e uma camada de saída. Uma das diferenças entre elas é de que suas camadas possuem formato bidimensional, ao contrário do MLP, que é unidimensional [Botalb et al. 2018]. As camadas ocultas podem realizar a convolução, a fim de realçar características, ou podem realizar o *pooling*, que reduz a amostra das características extraídas (tornando o processamento mais rápido) [Desai and Shah 2021].

2.2. Detecção de Faces

Para o reconhecimento de uma face, é necessário primeiramente que o algoritmo detecte a existência de uma face na imagem (ou quadro de um vídeo). O *dataset* WIDER FACE [Yang et al. 2016] é utilizado por diversos trabalhos para a realização de *benchmarks* relacionados à detecção de faces. Em sua página de resultados[5], encontram-se diversas submissões de trabalhos e comparações de seus resultados, nas diferentes dificuldades providas pelo *benchmark*.

Segundo [Kumar et al. 2019], há - de maneira geral - duas abordagens para a detecção de faces a partir de uma imagem: abordagem baseada em características e a abordagem baseada em imagem. A abordagem baseada em características consiste em extrair características da imagem que combinem com as características faciais conhecidas. Por outro lado, a abordagem baseada em imagem tenta obter a melhor correspondência entre imagens de treinamento e de teste. Encontram-se na Figura 1 algumas das diferentes técnicas para a detecção de faces, organizadas de acordo com as abordagens citas anteriormente.

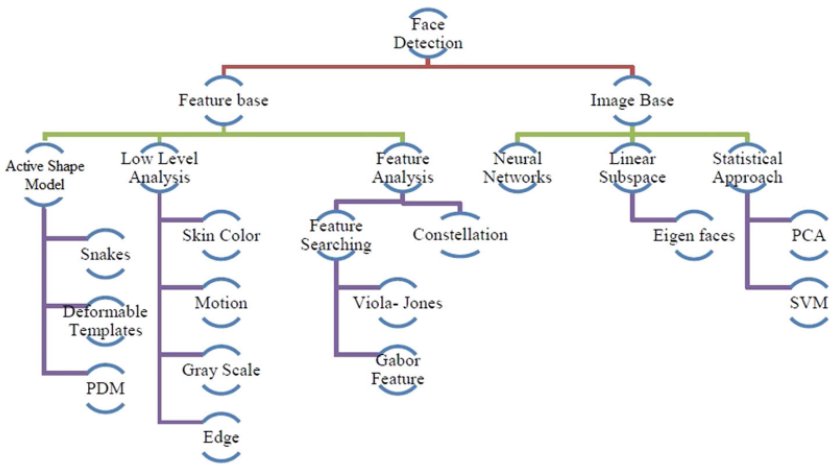


Figura 1. Diferentes técnicas para a detecção de faces [Kumar et al. 2019]

Abaixo encontra-se um breve resumo de alguns dos métodos mais relevantes para a detecção de faces em imagens.

[5] http://shuoyang1213.me/WIDERFACE/WiderFace_Results.html

2.2.1. Dlib

Dlib é um *toolkit* escrito em C++ para aplicações de aprendizado de máquina e análise de dados. É bastante utilizado na detecção de faces e na detecção de pontos de referência faciais (*landmarks*), pois pode estimar 68 pares de *landmarks* faciais [Tran 2021].

- **Dlib-HOG:** é um método de extração de características usando *Histogram of Oriented Gradients* (HOG) e *Support Vector Machine* (SVM). Esse método não obtem sucesso com poses muito variadas, funcionando apenas com faces frontais, ou levemente rotacionadas [Tran 2021]. Seu funcionamento basicamente se dá pela extração das características para um vetor que servirá de entrada para um algoritmo SVM, que então avaliará se há ou não uma face em determinada região. Essas características extraídas são histogramas das direções dos gradientes da imagem [Jadhav et al. 2021].
- **Dlib-CNN:** é a combinação de uma Rede Neural Convolucional, ou em inglês *Convolutional Neural Network* (CNN) com um *Maximum-Margin Object Detector* (MMOD). CNN é um algoritmo de aprendizado profundo que é usado para analisar imagens dadas como entrada [Jadhav et al. 2021].

2.2.2. YOLO

You Only Look Once (YOLO) é um método para a detecção de objetos utilizando aprendizado profundo, descrito por [Redmon et al. 2016]. A partir de uma imagem como entrada, gera as regiões que delimitam o objeto detectado, retornando informações como a altura, largura, centro e rótulo, além da confiança de previsão de cada uma das regiões. Faz isso utilizando apenas uma única rede neural [Tran 2021].

- **YOLOv5 e YOLO5Face:** nos anos seguintes à publicação original do YOLO [Redmon et al. 2016], diversas atualizações foram publicadas, uma vez que os autores originais não tiveram interesse em dar continuidade à pesquisa. YOLOv5[6] é a atualização mais recente aceita pela comunidade, apesar de não possuir um artigo publicado [Qi et al. 2021]. Sendo o YOLO um detector de objetos, [Qi et al. 2021] realizaram uma série de modificações e desenvolveram o YOLO5Face, com o intuito de especializar para a detecção de faces. Nos testes realizados pelos autores, concluiu-se que tanto o modelo maior (YOLOv5l6) quanto o modelo menor (YOLOv5n) alcançam desempenho próximo ou superior ao estado da arte nos subconjuntos fácil, médio e difícil de validação do WiderFace.

2.3. Reconhecimento de Faces

O reconhecimento de faces consiste em etiquetar cada uma das faces detectadas em uma imagem. No trabalho de [Kortli et al. 2020], os métodos de reconhecimento de faces são classificados em três diferentes abordagens: local, holística e híbrida. Essa classificações podem ser observadas na Figura 2.

Este trabalho apresenta alguns métodos utilizados para o reconhecimento de faces.

[6] `https://github.com/ultralytics/yolov5`

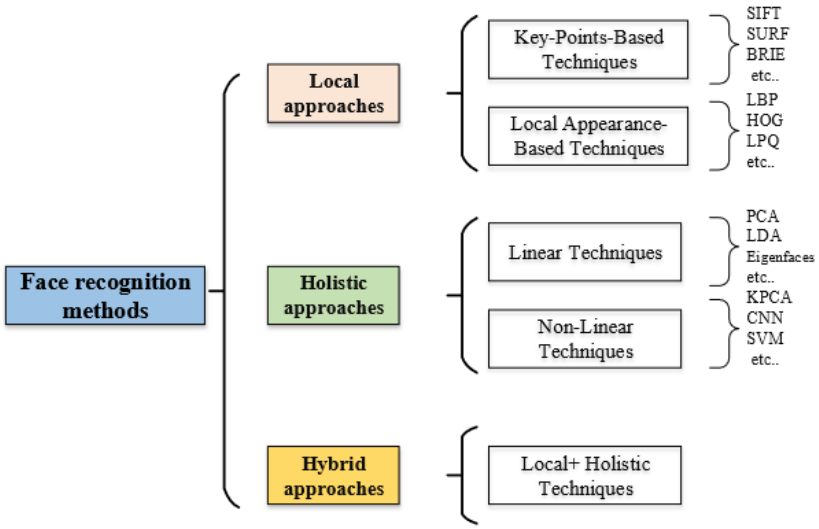


Figura 2. Classificação de métodos de reconhecimento de faces [Kortli et al. 2020]

2.3.1. ArcFace

Em sua publicação, [Deng et al. 2019] descreve que um dos principais desafios ao utilizar *Deep Convolutional Neural Networks* (DCNNs) para reconhecimento de faces em larga escala é o projeto de uma função de perda apropriada. O trabalho propõe uma função de perda chamada *Additive Angular Margin Loss* (ArcFace), que visa melhorar o poder discriminativo do modelo de reconhecimento facial. Dessa forma, substitui-se a função de perda tradicional *softmax*, uma vez que o tamanho da matriz de transformação linear aumenta linearmente com a quantidade de identidades [Tran 2021].

2.3.2. SphereFace

[Liu et al. 2017] introduz uma função de perda *softmax* angular (A-Softmax). O objetivo é obter uma distância intraclasse máxima menor do que a distância interclasse mínima, que permite as redes neurais convolucionais (CNNs) aprender características discriminativas angulares.

2.4. Anonimização de Faces

Há diversas formas de anonimizar faces. A mais simples consiste em embaçar os rostos desejados. Caso seja desejado preservar as características faciais, ao mesmo tempo que a identidade seja anonimizada, é possível utilizar uma abordagem de ofuscação de atributos.

2.4.1. Embaçamento da Face

[Pulfer 2019] comparou, em seu estudo, três métodos de processamento de imagem para determinar qual método é mais efetivo em reduzir a possibilidade de detecção de faces. Para tal, o autor implementou cada um dos métodos selecionados, com diferentes

parâmetros, e compararam a quantidade de faces detectadas nas imagens processadas com as imagens originais. Os métodos testados foram:

- ***Box Blurring:*** é uma forma de filtro *low-pass*. Utiliza valores uniformes no *kernel*, por isso é relativamente simples de calcular comparado a outros métodos, e portanto mais rápido;
- ***Gaussian Blurring:*** também é uma forma de filtro *low-pass*, porém utiliza valores derivados de uma função Gaussiana;
- ***Differential Privacy Blurring:*** esse método duplica a cor de cada *pixel* nos *pixels* vizinhos, de acordo com o tamanho de bloco desejado, e, além disso, também adiciona um ruído à cor de cada bloco, sendo este ruído aleatório de acordo com a distribuição de Laplace.

Dos três métodos, o *Differential Privacy* foi o mais efetivo em negar a detecção de faces nas imagens processadas. *Box Blurring* foi mais efetivo do que o *Gaussian Blurring*, que teve a menor efetividade dentre os três.

2.4.2. Ofuscação de Atributos

O método proposto por [Li et al. 2021] visa, a partir de uma face, gerar uma nova face correspondente. Esta nova face deve possuir uma aparência diferente da original, ainda preservando características relacionadas. O trabalho seleciona partes sensíveis da face, que são chaves para o processo cognitivo de reconhecimento. Estas partes são cabelo, sobrancelhas, olhos, nariz e lábios. O estudo analisa a modificação de diferentes números de atributos. [Li et al. 2021] concluem que o método proposto é capaz de preservar a identificação de faces, ao mesmo tempo que anonimiza a aparência facial.

3. Trabalhos Relacionados

Este capítulo apresenta dois trabalhos para a solução de problemas que, de alguma forma, são semelhantes ao abordado neste trabalho. Inicialmente, descreve-se uma tese que desenvolve técnicas variadas para acelerar o reconhecimento das faces em vídeos [Tran 2021]. Na sequência, discorre-se um trabalho que foca na proteção da privacidade em vídeos de transmissão ao vivo [Zhou and Pun 2020].

Para encontrá-los foi utilizada o Google Scholar[7], uma ferramenta para a pesquisa de literatura acadêmica e jornais científicos. Alguns termos utilizados na busca foram *"face detection"*, *"face recognition"* e *"face anonymization"*. Dentre os resultados, foram selecionados os artigos mais relevantes, sendo um dos fatores sua similaridade com a proposta deste trabalho.

3.1. *Fast Video-based Face Recognition in Collaborative Learning Environments*

Este trabalho tem como objetivo desenvolver um método de reconhecimento de faces em vídeo para reconhecer estudantes independente de suas poses. O método desenvolvido utiliza a abordagem *k-means* para identificar agrupamentos de imagens a fim de reconhecer faces em diferentes poses. Depois aplica a otimização multiobjetivo para estudar a

[7] scholar.google.com/

interdependência entre a taxa de reconhecimento, o número de *clusters* e a precisão do reconhecimento [Tran 2021].

Este trabalho utiliza o próprio vídeo de entrada para a geração dos protótipos de face de cada pessoa identificada. Protótipos estes que, mais tarde, serão utilizados para reconhecer, quadro a quadro do vídeo, quais estudantes estão interagindo entre si. Buscando acelerar e precisar o reconhecimento das faces, o autor analisou diferentes técnicas (e suas combinações) no processamento das faces detectadas. Ao todo, foram propostos quatro métodos que aplicam as técnicas, e também uma base de referência em que nenhuma dessas técnicas é aplicada. As técnicas analisadas foram:

- ***Data Augmentation*:** em alguns casos, diversas pessoas não aparecem com frequência, ou não se movem o suficiente para a obtenção de poses variadas. Essa técnica tenta resolver esses problemas ao criar, virtualmente, diferentes poses a partir de uma única imagem da face, aumentando então a variedade de poses. Para isso, aplicam-se a combinação de algumas edições na imagem: espelhamento horizontal, rotação, alteração da escala, movimentação horizontal e vertical, e transformação de cisalhamento.
- ***K-means Clustering*:** na base de referência, todas as faces de cada participante foram utilizadas para criar uma coleção de protótipos associados com os participantes. Porém, cada um possuía aproximadamente 9.500 imagens, o que levaria muito tempo para processar. Portanto, foi introduzida a técnica *K-means Clustering*. Essa técnica consiste em utilizar o algoritmo *K-means* para agrupar faces muito semelhantes, quando a pessoa não se move muito entre um quadro e outro do vídeo, por exemplo. Dessa forma, o trabalho espera que os centróides resultantes do *K-means* representem as diferentes poses de cada participante.
- ***Sparse Sampling*:** essa técnica consiste em realizar uma amostragem esparsa dos quadros do vídeo, no lugar de utilizar todos os quadros. Por exemplo, o algoritmo apresentado utiliza apenas um quadro por segundo do vídeo para alcançar a escassez.
- ***Frames Skipping*:** para acelerar o processamento, algumas opções de pulos de quadros foram testadas: sem pulos, 5, 10, 15, 20, 30 e 60 quadros.

Para a base de referência, foi treinado e testado um modelo de classificador SVM usando o código disponibilizado pelo scikit-learn[8]. Além deste, outros oito classificadores foram implementados a fim de comparar os resultados entre estes modelos, sendo o SVM obtendo o melhor resultado.

Tendo os resultados para uma base de referência, seguiram-se os testes dos métodos propostos no trabalho. Concluiu-se que o método final e otimizado resultou em um reconhecimento mais rápido e com melhora na acurácia do reconhecimento de faces. O método proposto atingiu uma acurácia de 71,8%, enquanto o modelo de referência teve 62,3%, enquanto rodou mais rápido que todos os outros métodos propostos.

3.2. *Personal Privacy Protection via Irrelevant Faces Tracking and Pixelation in Video Live Streaming*

Neste trabalho, [Zhou and Pun 2020] afirmam que com a popularização das plataformas de *streamings* de vídeo, tornou-se preocupante a privacidade das pessoas contidas nos

[8]`https://scikit-learn.org/stable/modules/svm.html`

vídeos. Os autores reiteram que é inviável para essas plataformas de *streaming* alocar esforço humano para executar as anonimizações necessárias. Citam que essas plataformas, incluindo YouTube e Microsoft Azure, focam no processamento *offline* dos vídeos, deixando o campo das transmissões ao vivo inexplorado. Portanto, neste artigo apresenta-se um novo método desenvolvido chamado de *Face Pixelation in Video Live Streaming* (FPVLS), que automaticamente anonimiza as pessoas em *streaming de vídeos*.

Segundo os autores, aplicar somente rastreadores de face acarreta em diversos problemas, como desvio do alvo, eficiência computacional, e superpixelização. Portanto, para realizar a pixelização de maneira rápida e precisa das faces irrelevantes, o FPVLS é organizado em uma estrutura de dois estágios principais. Em quadros individuais, o FPVLS utiliza detecção de rosto baseada em imagem e *Embedding Networks* para produzir vetores de face. No estágio de geração de trajetória, o algoritmo *Positioned Incremental Affinity Propagation* (PIAP) proposto aproveita os vetores de face e informações de posição para associar rapidamente os rostos da mesma pessoa em diferentes quadros. Porém, essas informações de trajetórias possivelmente serão intermitentes e não confiáveis, portanto introduziu-se uma etapa de refinamento de trajetória, mesclando uma rede proposta com um teste entre duas amostras, baseado na estatística *Empirical Likelihood Ratio* (ELR). Então, aplica-se um filtro gaussiano nas trajetórias antes da pixelização final.

No artigo encontra-se uma figura comparativa entre *frames* do vídeo original, *frames* do mesmo vídeo processado pela ferramenta do YouTube, e também processado pelo FPVLS. É possível encontrar os problemas citados pelos autores, como o desvio do alvo e a superpixelização. Também citam métodos que inicialmente tentaram implementar: 3D CNN, MOT, MFT. Porém encontraram os problemas citados anteriormente.

Considerando os prós e contras dos métodos testados, desenvolveram a FPVLS. Junto com um estágio de pré-processamento e um estágio de pós-processamento, o FPVLS possui dois estágios principais: o estágio de geração de trajetórias de face e o estágio de refinamento de trajetória.

Para a realizar a experimentação, foi necessário coletar diversos vídeos de transmissão ao vivo das plataformas YouTube e Facebook, e construir um *dataset*, uma vez que não há disponível nem um *dataset*, nem um *benchmark*. Por ser o primeiro trabalho realizado no campo de pixelização de vídeos de transmissão ao vivo, os autores não encontraram métodos de comparação, portanto utilizaram os potenciais algoritmos e os adaptaram para a tarefa de pixelização de face. Para as métricas, utilizaram as já amplamente aceitas no campo de rastreamento, e propuseram as seguintes métricas para indicar a performance do algoritmo: *Multi-Face Pixelation Accuracy* (MFPA), *Multi-Face Pixelation Precision* (MFPP), e *Most Pixelated* (MP). Além disso, *Over-Pixelation Ratio* (OPR) e a capacidade de lidar com o número de pessoas *Number of People* (NoP) são as outras duas métricas personalizadas para a tarefa de pixelização.

No geral, o FPVLS obtém melhor desempenho em MFPA, MFPP, MP e OPR em todos os subconjuntos de dados. O método proposto aumenta notavelmente a métrica MP em todo o conjunto de dados, indicando que o FPVLS produz uma pixelização mais duradoura, uma vez que esta métrica está relacionada à quantidade de frames consecutivos que foram pixelizados corretamente.

3.3. *An architecture for automatic multimodal video data anonymization to ensure data protection*

Neste trabalho, [Grosselfinger et al. 2019] projetaram e implementaram um *pipeline* de anonimização de vídeos (anonimização de faces e placas de carro). Este *pipeline* contém *plugins* para ler, modificar e gravar diferentes formatos de imagem, bem como métodos para detectar as regiões que devem ser anonimizadas. Isso inclui um método para determinar as posições da cabeça e um detector de objetos para as placas, ambos baseados em métodos de aprendizado profundo de última geração.

Para a detecção das faces é estimado primeiramente a pose do corpo da pessoa, utilizando OpenPose[9]. O retorno de OpenPose inclui seis pontos da região facial, e a partir desses pontos é possível determinar o centro da face. O tamanho da região facial é determinado a partir das distâncias entre esses pontos.

As regiões a serem anonimizadas definidas são, então, salvas em um arquivo XML, onde é possível realizar algum refinamento manual, caso necessário. Por exemplo, eliminar falso positivos, adicionar ou alterar regiões detectadas. A anonimização é realizada com uma versão modificada do *gaussian blur*, tratado neste trabalho em 2.4.1. Segundo os autores, o processo de anonimização atinge desempenho quase humano, sem precisar de praticamente nenhum trabalho humano.

3.4. Comparativo

Cada um desses trabalhos contribui de alguma forma para a realização deste.

O primeiro trabalho é o que mais se assemelha à proposta deste. Todas as etapas a serem desenvolvidas neste trabalho são, na medida do possível, bastante semelhantes às etapas da ferramenta a ser desenvolvida. A etapa de reconhecimento, porém, se difere um pouco, uma vez que na ferramenta proposta o usuário irá entrar com as imagens para a geração dos protótipos de face.

O segundo trabalho apresenta uma proposta de rastreamento das faces, que poderia ser uma adição bastante interessante para a ferramenta proposta neste trabalho.

O terceiro trabalho realiza a anonimização, além de faces, de placas de carro, o que não é o foco deste trabalho. Porém, utiliza do salvamento das regiões detectadas em um arquivo, permitindo - dessa forma - que o usuário faça ajustes antes do processamento final. Esta também é uma adição a ser considerada para a ferramenta proposta.

4. Desenvolvimento

Este capítulo apresenta as ferramentas utilizadas e o desenvolvimento do *framework* proposto.

4.1. Premissas e Decisões

Para o desenvolvimento deste projeto, algumas escolhas relacionadas à tecnologias e premissas de funcionamento foram feitas, e encontram-se abaixo:

[9][Cao et al. 2019]

4.1.1. Entradas e Saídas

Este trabalho parte da premissa que serão fornecidos:

- Uma lista com os nomes das pessoas a serem mantidas;
- Exemplares das faces das pessoas a serem mantidas, sejam eles imagens ou vídeos;
- Vídeo a ser processado.

E terá de saída o arquivo de vídeo com as faces desejadas anonimizadas.

4.1.2. Detecção e Reconhecimento de Faces

Em relação à detecção e reconhecimento de faces, optou-se por utilizar a biblioteca InsightFace. Esta escolha visou a simplificação do desenvolvimento, uma vez que ambas as funções (detecção de face e geração dos *embeddings*) já são implementadas, evitando a necessidade de múltiplas bibliotecas. Será discutida com mais profundidade posteriormente neste trabalho.

4.1.3. Anonimização

Conforme discutido em 2.4, existem diversas formas para anonimizar uma face. Neste trabalho o autor decidiu por realizar um filtro do tipo *Gaussian Blur*. Este filtro é amplamente conhecido, e já se encontra implementado na biblioteca utilizada para manipulação de imagens, OpenCV.

Para os autores deste trabalho, o método de anonimização através da ofuscação de atributos seria o ideal, porém aumentaria consideravelmente a complexidade de implementação. Por outro lado, preencher a região da face a ser anonimizada com *pixels* seria o método mais simples, mas não seria agradável àqueles espectadores do vídeo processado pelo *framework* proposto. Os autores consideram o *Gaussian Blur* um método que junta as vantagens de cada um desses outros dois métodos.

4.2. Ferramentas

Nesta seção discutem-se algumas das ferramentas utilizadas durante a implementação do trabalho.

4.2.1. InsightFace

InsightFace[10] é uma biblioteca para Python que implementa uma diversidade de algoritmos relacionados à análise de face, como reconhecimento, detecção e alinhamento. Nela é possível encontrar diversos modelos já pré-treinados, que aceleram e facilitam o desenvolvimento do projeto.

[10] `https://insightface.ai/`

4.2.2. Keras e Tensorflow

Keras[11] é uma interface de código aberto que facilita a utilização da biblioteca, também de código aberto, Tensorflow[12]. Com elas pôde-se facilmente implementar redes neurais com estruturas variadas durante o desenvolvimento e testes do projeto.

4.2.3. Scikit-learn

Scikit-learn[13] é uma biblioteca de *machine learning* para Python, e foi utilizada durante o treinamento do MLP desenvolvido.

4.2.4. OpenCV

OpenCV[14] é uma biblioteca para manipulação de imagens, e é desenvolvida principalmente em linguagem C++. Para utilizá-la neste projeto, fez-se o uso da biblioteca *opencv-python*. OpenCV foi utilizada neste projeto com a finalidade de criar um filtro de embaçamento na região da face detectada.

4.3. Projeto do programa

Para fins didáticos este trabalho pode ser abstraído em três etapas:

- Aprender a reconhecer as pessoas desejadas;
- Detectar e classificar, quadro a quadro, todas as faces presentes no vídeo;
- Anonimizar as faces desejadas.

Na figura abaixo, encontra-se a relação das três etapas citadas anteriormente, com as três principais classes do código desenvolvido:

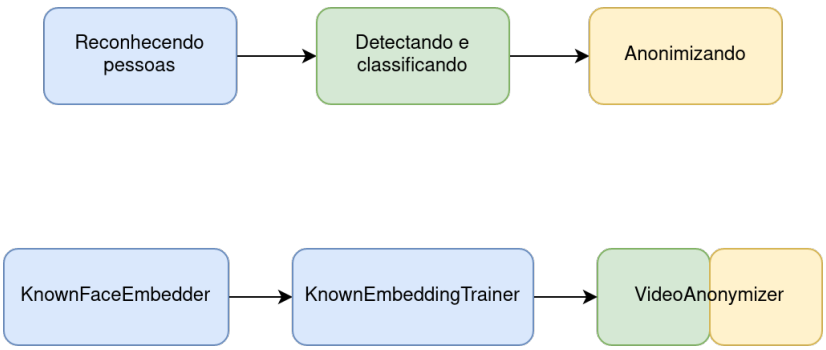


Figura 3. Abstração do projeto do programa e estrutura do código desenvolvido

Essas etapas são encontradas nas subseções abaixo:

[11] https://keras.io/
[12] https://www.tensorflow.org/
[13] https://scikit-learn.org
[14] https://opencv.org/

4.3.1. Reconhecendo pessoas

Para cada uma das pessoas fornecidas, gera-se com a InsightFace os *embeddings* de todas as imagens, ou *frames* dos vídeos. Cada um desses *embeddings* é rotulado com o nome da pessoa, e finalmente salva-se o resultado em um arquivo.

A partir dos *embeddings* rotulados treina-se um MLP capaz de classificar - com um grau de confiança atribuído - a quem pertence aquela face.

4.3.2. Detectando e classificando

Com o MLP treinado, dá-se início à segunda etapa, onde são detectadas e classificadas todas as faces presentes no vídeo. Para tal, processa-se um *frame* de cada vez do vídeo, e utiliza-se novamente da InsightFace para detectar as faces e gerar seus respectivos *embeddings*. Então, para cada um desses *embeddings*, alimenta-se o MLP e verifica qual a pessoa identificada (isto é, qual das classes possui a maior confiança).

Enfim, as informações de cada face: região, pessoa identificada e grau de confiança (que vai de 0 a 1), são salvas em um arquivo, e serão utilizadas na próxima etapa.

4.3.3. Anonimizando

Por último, o vídeo é processado novamente do início, utilizando as faces detectadas na etapa anterior. Caso a confiança ultrapasse o limiar configurado, tem-se certeza de que aquela face detectada realmente pertence àquela pessoa.

Caso não haja certeza que a face pertence à pessoa a ser mantida, ou a pessoa identificada não está inclusa na lista de pessoas a serem mantidas, ou seja, deve ser anonimizada, utiliza-se a região detectada para criar um filtro do tipo *Gaussian Blur*.

4.4. Código desenvolvido

A partir da estrutura abstraída do programa, pode-se compreender melhor o código desenvolvido.

4.4.1. Geração dos *embeddings*

A classe desenvolvida responsável pela geração dos *embeddings* foi chamada de **KnownFaceEmbedder**. Tem-se como entrada as imagens que possuem as faces, rotuladas através da estrutura de diretórios que se organizam. A partir disso, para cada imagem é gerado um vetor de *floats*, isto é, o *embedding* da face presente naquela imagem. Então, é realizado o salvamento de um arquivo composto pelo mapeamento de um rótulo com um *embedding*. A Figura 4 apresenta um diagrama das entradas e saídas da classe KnownFaceEmbedder.

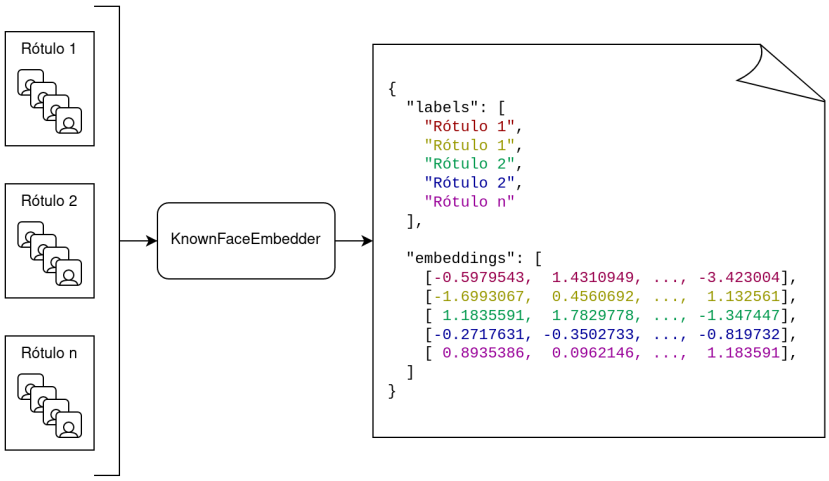


Figura 4. Funcionamento da classe responsável pela geração dos *embeddings*

4.4.2. Aprendizado

A classe responsável por treinar a rede neural para a classificação das faces foi chamada de **KnownEmbeddingTrainer**. O arquivo gerado contendo os rótulos e os *embeddings* é utilizado aqui para realizar o treinamento. A Figura 5 apresenta um diagrama das entradas e saídas da classe KnownEmbeddingTrainer.

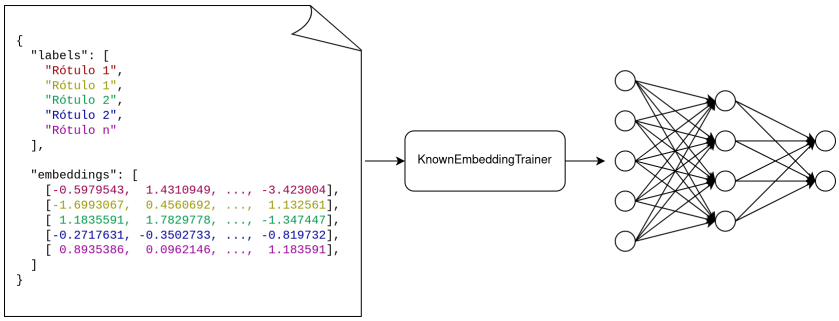


Figura 5. Funcionamento da classe responsável pela geração dos *embeddings*

4.4.3. Anonimização do vídeo

Por fim, a classe responsável por anonimizar as faces desejadas foi chamada de **VideoAnonymizer**. Para anonimizar o vídeo, esta classe recebe como entrada o arquivo de vídeo a ser processado, o MLP treinado anteriormente, e uma lista de nomes (rótulos) das pessoas a terem sua identidade mantida no vídeo. Aqui também é utilizado o InsightFace para detectar as faces e gerar os *embeddings*, que serão utilizados como entrada do MLP para a classificação de cada uma das faces. Caso a face identificada estiver na lista de nomes, e possuir um grau de confiança maior que o configurado, esta não será anonimizada. Caso contrário, será anonimizada. Essa classe também salva em um arquivo as regiões das faces detectadas, junto com o resultado do MLP (rótulo e probabilidade). A

utilização desse arquivo será apresentada na seção 4.5. A Figura 6 apresenta um diagrama das entradas e saídas da classe VideoAnonymizer.

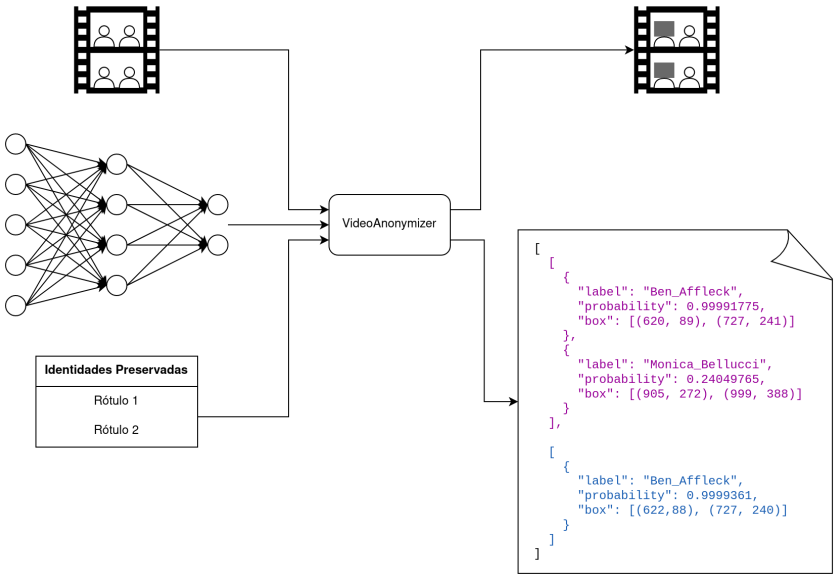


Figura 6. Funcionamento da classe responsável pela anonimização do vídeo

4.4.4. Uso final do programa pelo usuário

A utilização do programa desenvolvido neste trabalho se dá pela linha de comando do *shell*, ao invocar o Python.

Há um argumento obrigatório, que é o caminho para o arquivo de vídeo que será processado.

As imagens de cada uma das pessoas devem estar dentro de um diretório cujo nome será utilizado para rotulá-las posteriormente. Todos diretórios de pessoas devem estar dentro de outro diretório chamado ”datasets”. Dois exemplos de caminhos: `./datasets/Ben_Affleck/000001.jpg` e `./datasets/Michelle_Obama/000100.jpg`.

Caso o usuário queira definir o tamanho das camadas do MLP, pode fazê-lo através da *flag* `-l` (ou `--layers`). Se não o fizer, será utilizado o padrão (3 camadas): 256, 256 e 512.

Caso o usuário queira definir as épocas de treino do MLP, pode fazê-lo através da *flag* `-e` (ou `--epochs`). Se não o fizer, será utilizado o padrão: 1.

Naturalmente, também pode ser informado o nome das pessoas que terão suas faces mantidas (não anonimizadas) no vídeo. O usuário o faz através da *flag* `-pi` (ou `--preserved-identities`).

Um exemplo de utilização, em que se mantém apenas as faces do Ben Affleck:

```
python anonimify.py oscar.mp4 -pi Ben_Affleck
```

Outro exemplo de utilização, em que se mantém apenas as faces da Michelle Obama, configura-se duas camadas (tamanhos 1024 e 512 neurônios), e treina-se por 3 épocas:

```
python anonimify.py oscar.mp4 -pi Ben_Affleck
Michelle_Obama -l 1024 512 -e 3
```

4.5. Manipulação do reconhecimento das faces

Optou-se por avaliar a acurácia da anonimização através da comparação do resultado obtido em 4.4.3 com a rotulação manual. Para facilitar essa rotulação, desenvolveu-se um *software* que é capaz de ler o arquivo gerado após o processamento do vídeo, e mostrar ao usuário - quadro a quadro - o reconhecimento das faces. A partir disso, o usuário pode alterar o rótulo da pessoa reconhecida e seu grau de confiança, e por fim, salvar o arquivo.

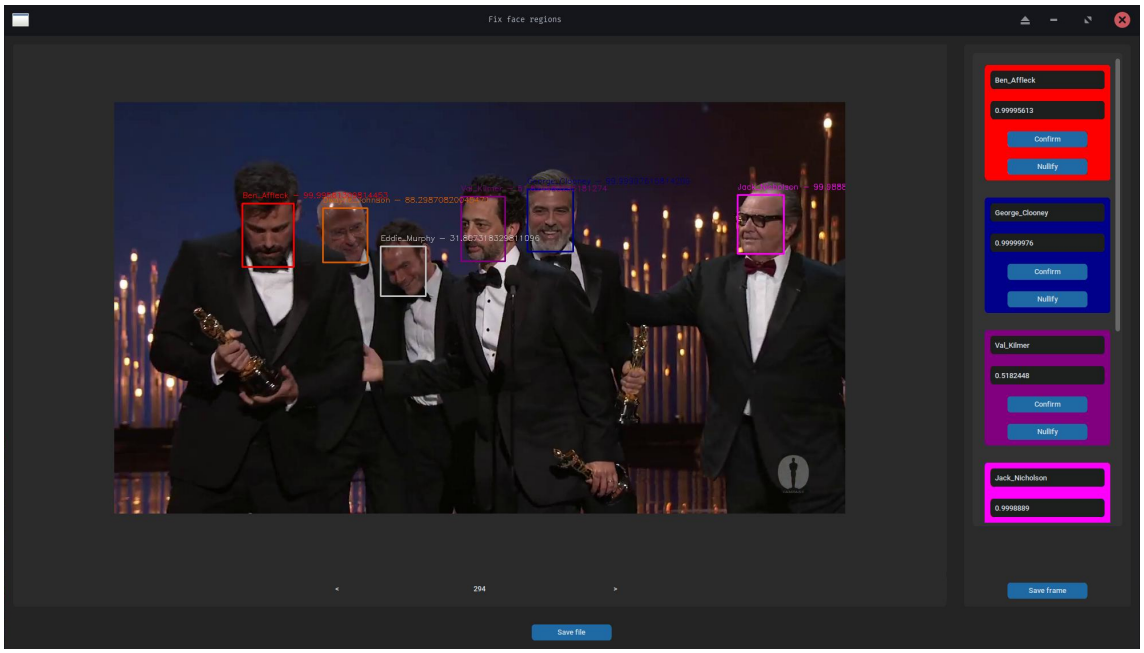


Figura 7. *Software* para manipulação do reconhecimento das faces

Posteriormente, na etapa de testes, este *software* será utilizado para comparar o arquivo gerado pelo modelo treinado com o arquivo corrigido.

5. Testes

Ao todo, 672 modelos de MLP foram treinados, utilizando mais de 15 mil fotos, contendo mais 160 identidades diferentes, com a finalidade de criar a ferramenta mais eficaz possível. Este capítulo discorrerá desses testes realizados.

5.1. Infraestrutura

Todos os testes foram realizados em um computador com distribuição Linux Manjaro 22.0.0, processador AMD Ryzen 5 3600X de 6 *cores* a 3,8 GHz, 16GB de RAM DDR4, e uma placa de vídeo NVIDIA GeForce RTX 2060 SUPER com 8GB de VRAM.

5.2. Conjunto de dados

5.2.1. Vídeo alvo

Os testes foram realizados em um um vídeo[15] encontrado na plataforma YouTube, que consiste em um trecho da premiação de cinema, em que o filme Argo recebe o prêmio de melhor filme, em 2013. O vídeo foi cortado com a finalidade de realizar diversos testes em um período de tempo menor (dez segundos; versão completa possui quase oito minutos).

Optou-se por esse vídeo, pois possui apenas pessoas públicas, das quais imagens são facilmente encontradas na *internet*, facilitando também a geração de um *dataset* que será utilizado para treinar o MLP.

Para todos os testes realizados nesse vídeo, utilizaram-se as seguintes pessoas a terem suas identidades mantidas no vídeo: Ben Affleck, George Clooney, Jack Nicholson e Michelle Obama. Essas quatro pessoas serão denominadas de *Preserved Identities* (PI, em português Identidades Preservadas). Do mesmo modo, as outras pessoas que farão parte do treino da rede neural, mas que não terão suas identidades mantidas em vídeo, serão chamadas por *Hidden Identities* (HI, em português Identidades Ocultas).

5.2.2. *Dataset* de treino para o MLP

Para a criação de um *dataset* foi criado um *script* em Python que utiliza a biblioteca *icrawler*[16] para baixar imagens de uma lista atores e pessoas públicas. Foi criada uma lista com 163 nomes, entre eles atores, atrizes e políticos norte americanos. Para cada uma dessas pessoas foram baixadas cerca de cem imagens.

5.3. Estrutura do MLP

Foi desenvolvido um *script*, também em Python, para a preparar o treino dos mais diversos MLPs, contendo quantidades de camadas, números de neurônios por camada oculta, e épocas de treino distintas.

5.4. Limiar de confiança do reconhecimento

Conforme descrito em 4.3.2 e 4.3.3, é necessário configurar um limiar de confiança. Todos os testes foram realizados com limiar configurado em 0,8, que pode ser interpretado como ”80% de confiança no reconhecimento”.

5.5. Avaliação da acurácia

Primeiramente, gerou-se com o *software* descrito em 4.5 um arquivo de reconhecimento manual do vídeo alvo. A comparação segue o seguinte algoritmo:

Caso seja uma face que deveria ser **mantida**: acertou o nome (a etiqueta) e o grau de confiança é maior que o configurado: +1 acerto. Caso contrário: +1 erro.

Caso seja uma face que deveria ser **anonimizada**: reconheceu sendo uma PI (etiquetou com um dos nomes cuja face deveria ser mantida), e o grau de confiança é maior que o configurado: +1 erro. Caso contrário: +1 acerto.

[15] `https://youtu.be/FtLKn5Y1ulc`
[16] `https://icrawler.readthedocs.io/`

Dessa forma, pontua quando há o correto reconhecimento, e não pontua caso a ferramenta mostre uma face que deveria ser anonimizada.

Enfim, a acurácia é calculada com:

$$acerto/(acerto + erro)$$

5.6. Resultados

5.6.1. Experimento

Foram treinadas todas as redes neurais que acomodassem todas as combinações de das seguintes características:

- número de camadas: de 1 a 3;
- número de neurônios por camada escondida: 256, 512, 1024 e 2048;
- épocas de treino: 2 e 3;
- quantidade de HI: 4, 8, 64, 128.

Dessa forma, a quantidade de testes realizados foi de 672, que demorou quase 13 horas de execução, e pode-se observar alguns comportamentos, que serão descritos abaixo.

5.6.2. Análise do impacto das características

Para encontrar se há variáveis que impactam mais ou menos a acurácia do modelo, esmiuçar-se-ão os diferentes modelos gerados, a partir de cada umas das características descritas.

5.6.3. Épocas de treino

Pode-se analisar visualmente na Figura 8 que, ao isolar apenas a quantidade de épocas de treino, essa variável parece não ser tão relevante no impacto na acurácia.

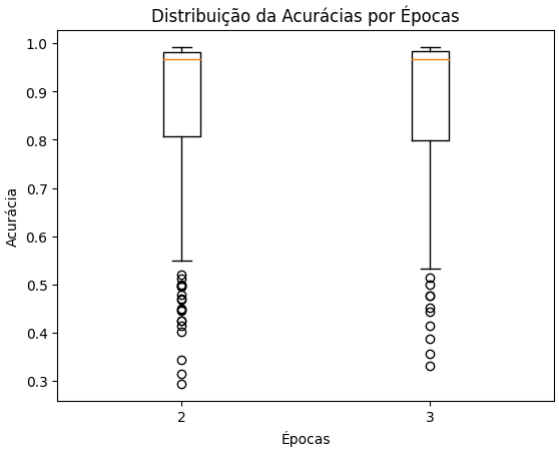


Figura 8. Diagrama de caixa relativo às épocas de treino

Número de camadas	Quantidade total	Quantidade no top 100	Proporção
1	32	8	25,00%
2	128	24	18,75%
3	512	68	13,28%

Tabela 1. Proporção das ocorrências de cada número de camadas no top 100

5.6.4. Quantidade de camadas

Diferentemente das épocas, a quantidade de camadas parece ser diretamente relacionada à acurácia, como pode ser analisado na Figura 9. Apesar das medianas estarem próximas, o limite inferior da caixa de apenas uma camada é maior, o que indica que a maior parte das redes contendo apenas uma camada possui, em geral, uma acurácia maior.

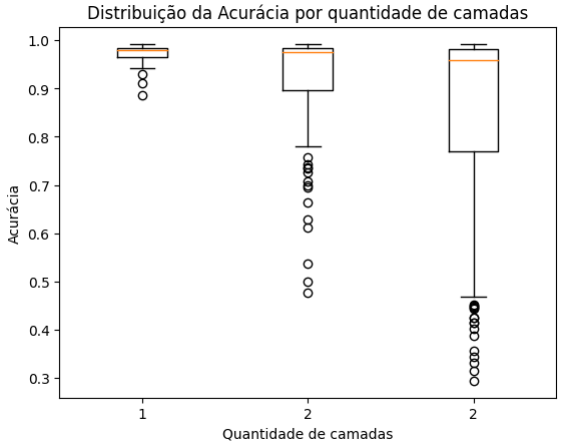


Figura 9. Diagrama de caixa relativo à quantidade de camadas

Devido à maneira que os testes foram elaborados, há mais testes com 3 camadas do que com 2 e 1, e mais testes com 2 camadas do que com 1, pode-se também analisar a proporção da ocorrências de cada quantidade no top 100 testes, ordenados pela acurácia. Dessa forma, tem-se o seguinte resultado:

Conforme avaliado na Tabela 1, a proporção de testes com apenas uma cada no top 100 é maior quando comparado com as demais quantidades.

5.6.5. Quantidade de neurônios

Dessa forma, pode-se avaliar o impacto da quantidade de neurônios quando há apenas uma única camada. A distribuição da acurácia, para cada caso, pode ser visualizado na Figura 10, onde é notável que os testes realizados com 1024 neurônios tendem a performar melhor.

5.6.6. Quantidade de pessoas conhecidas

Uma das hipóteses levantadas durante a construção deste trabalho, é a de que quanto mais pessoas a MLP souber reconhecer, mais eficaz será. Ou seja, quando se detectar um rosto

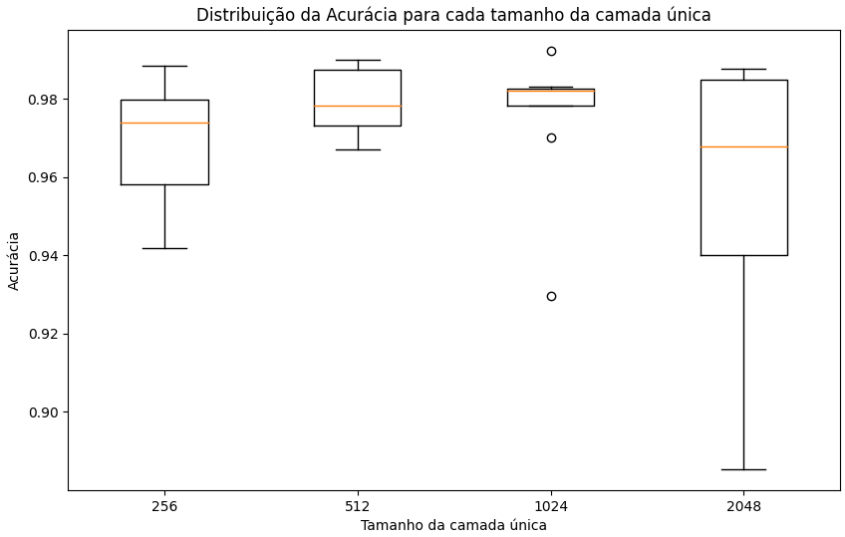


Figura 10. Diagrama de caixa relativo à quantidade de neurônios quando há uma única camada

conhecido, que a probabilidade de acerto seja alta. Por outro lado, quando detectar um rosto que não conhece, apesar de reconhecer como alguma das pessoas contidas no *dataset* de treino (como é a natureza de um MLP), a probabilidade seja o mais baixa possível, de preferência abaixo do *threshold* configurado.

Dessa forma, ao analisar a distribuição da acurácia para cada quantidade de HI (Figura 11), além das quatro PI, pode-se notar que há uma melhora significativa devida ao aumento de identidades no treino.

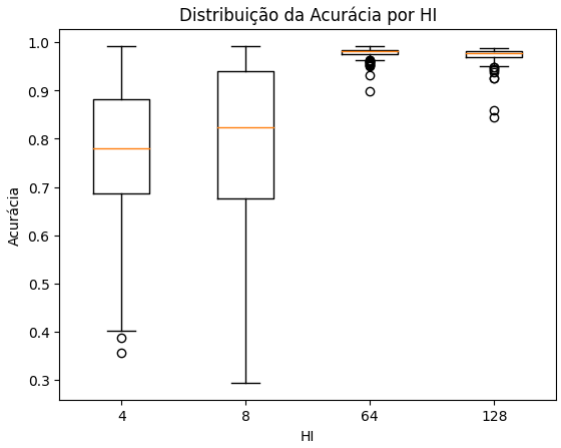


Figura 11. Diagrama de caixa relativo à quantidade de HI

6. Análise dos Resultados

Todos os modelos com acurácia acima de 99%, podem ser encontrados na Tabela 2.

A mediana da acurácia de todos os modelos treinados é de aproximadamente 96,7%, ou seja, a grande maioria dos modelos performou muito bem. Além disso, o melhor modelo (chamado de ”1024_3_4PI_8HI”) foi capaz de acertar 1299 reconhecimentos,

Camada 1	Camada 2	Camada 3	Épocas	Quantidade de HI	Acurácia
1024			3	8	99,24%
256	256		2	4	99,16%
2048	2048	512	3	64	99,16%
256	256		2	8	99,08%
256	1024	256	3	64	99,01%
256	1024		2	8	99,01%
2048	256		3	64	99,01%
2048	512	512	3	64	99,01%
2048	512		3	64	99,01%
256	512	1024	2	8	99,01%
256	256	256	3	8	99,01%
512			2	8	99,01%

Tabela 2. Proporção das ocorrências de cada número de camadas no top 100

de um total de 1309 faces detectadas no vídeo testado, correspondendo a uma acurácia de 99,24%. Na Figura 12 encontram-se dois exemplos onde as quatro faces desejadas foram corretamente reconhecidas.

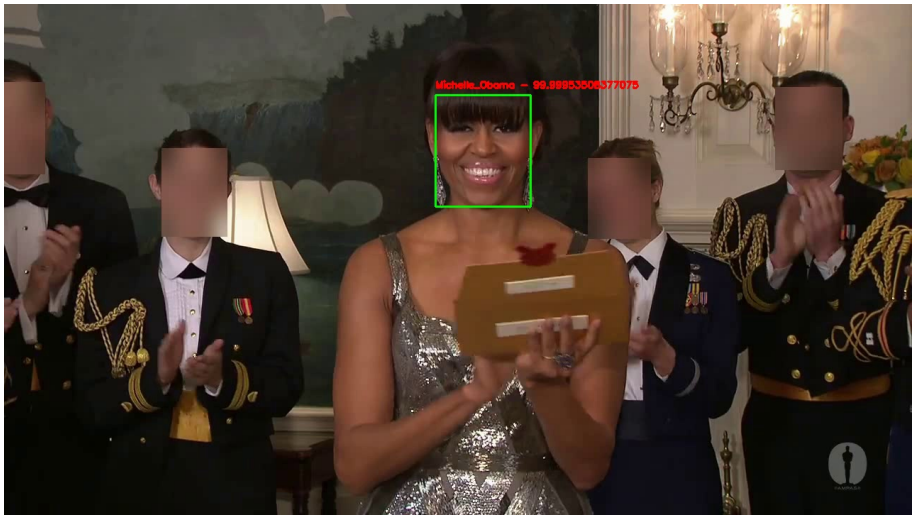
Na Figura 13, encontram-se dois quadros do vídeo alvo após o processo de anonimização. É possível ver que ambos Ben Affleck e George Clooney foram reconhecidos no quadro 217. Porém, à medida que o George Clooney se movimenta para trás do Jack Nicholson, seu rosto fica obstruído, e a rede se confunde, não mais o reconhecendo.

Um ser humano é capaz de compreender o contexto do movimento que o George Clooney está realizando. Dessa forma, mesmo que a face detectada seja a da Figura 14 (bastante obstruída), somos capazes de reconhecer como sendo a mesma pessoa.

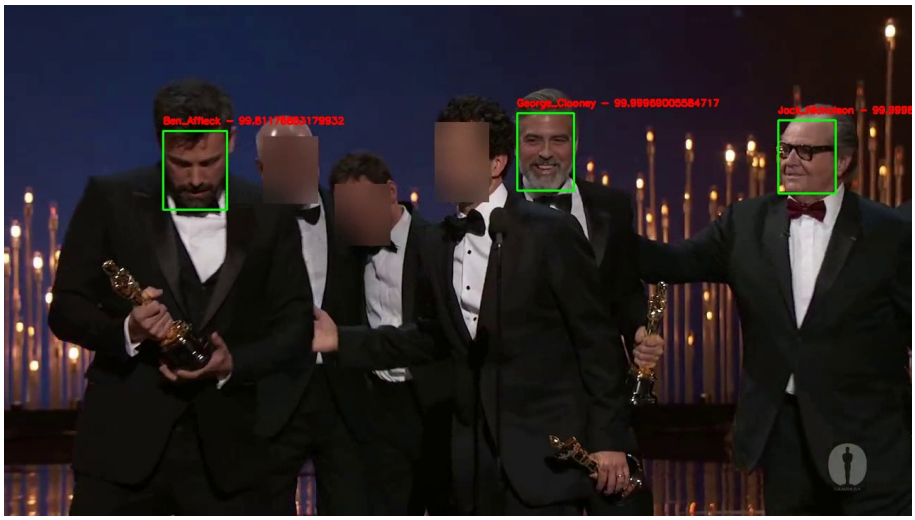
Vale destacar que, nesse caso específico, a ferramenta optou por anonimizar essa face não por a reconhecer erroneamente, mas sim por falta de confiança. Mais profundamente, no quadro anterior (217), a rede reconheceu sendo o George Clooney com uma probabilidade acima de 99%, enquanto no quadro seguinte (Figura 14) também a reconheceu sendo o George Clooney, porém com a probabilidade de apenas 67% (abaixo do limiar configurado), portanto a anonimizando.

Outro destaque importante é que todos os dez erros foram por anonimizar uma face que deveria ser mantida, e não por mostrar uma face que deveria ser anonimizada. Dessa forma, a ferramenta não deixou de cumprir o seu propósito.

Ainda assim, há bastante margem para mais testes, que podem incluir outros vídeos, de naturezas distintas, e o aumento de combinações de variáveis, como por exemplo mais tempo de treino (número maior de épocas). Realizar experimentos com outros vídeos serviria para confirmar que os resultados encontrados para um vídeo, replicam-se para outros. Não podemos afirmar que o melhor modelo encontrado nos experimentos



(a) Michelle Obama reconhecida



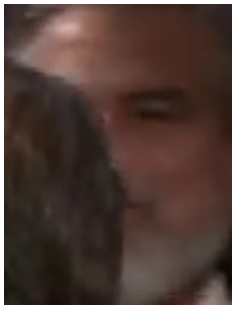
(b) Ben Affleck, Jack Nicholson e George Clooney reconhecidos

Figura 12. Demonstração do correto reconhecimento das pessoas desejadas

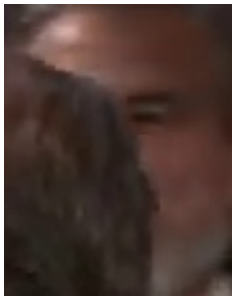
(a) Quadro 217/300

(b) Quadro 218/300

Figura 13. Comparação dos quadros onde há falha do reconhecimento



(a) Face do George Clooney (quadro 217)



(b) Face do George Clooney (quadro 218)

Figura 14. Comparação das faces onde há falha do reconhecimento

deste trabalho será também o melhor modelo para outro vídeo.

7. Conclusão

7.1. Considerações Finais

Dados os resultados apresentados no capítulo anterior, concluiu-se que a ferramenta desenvolvida atingiu satisfatoriamente o resultado esperado. Que no melhor dos casos anonimizou todas as faces que deveria, necessitando de pouquíssimo trabalho de ajuste para atingir a perfeição. Foi possível perceber que algumas configurações de MLP tendem a gerar redes mais precisas, porém com a mediana da acurácia em 96,7%, pode-se afirmar que essa técnica é bem flexível e pode ser utilizada para a finalidade deste trabalho. Para afirmações mais específicas sobre as diferentes configurações de MLP, mais testes são necessários, como, por exemplo, verificar se a acurácia de uma rede em um vídeo se mantém para outro vídeo. Espera-se que este trabalho sirva de inspiração para que demais trabalhos sejam desenvolvidos nessa área, uma vez que a privacidade é um bem muito importante, e provamos que é possível desenvolver uma automação eficaz, reduzindo consideravelmente o esforço manual humano.

7.2. Trabalhos Futuros

Durante todo o processo de desenvolvimento deste trabalho, surgiram diversas ideias para trabalhos futuros. Ideias essas que envolvem ampliar a aplicabilidade deste trabalho, melhorar sua performance, ou facilitar seu uso. Alguns possíveis trabalhos futuros encontram-se listados abaixo.

7.2.1. Interface gráfica

Para facilitar o uso de pessoas com nenhum conhecimento em programação, seria de bastante importância a implementação de uma interface gráfica amigável e fácil de usar, que permitisse aos usuários mais avançados a possibilidade de ajustar parâmetros.

Outra funcionalidade possível é a edição das regiões de faces detectadas e reconhecidas. Essa edição permitiria o usuário facilmente remover falsos negativos/positivos. Ressalta-se que essas regiões já são salvas em um arquivo separadamente, o que facilitaria a implementação dessa funcionalidade.

7.2.2. Análise de contexto

Ao processar cada *frame* individualmente, ocasiona-se um dos principais motivos de falha deste trabalho, que é a falta de entendimento de contexto entre *frames*, isto é, saber que a pessoa detectada no *frame* atual é a mesma do *frame* anterior e seguinte. Este é um problema, pois, tratando-se de vídeos, podem ocorrer embaçamento da face devido ao movimento da pessoa ou câmera (*motion blur*), ou até mesmo a pessoa mover a cabeça em um ângulo em que seja virtualmente impossível reconhecê-la, mesmo para um humano familiar com aquela pessoa.

Um possível trabalho futuro seria adicionar a questão contexto ao *pipeline* deste trabalho.

7.2.3. Anonimização com ofuscação de atributos

Conforme citado anteriormente neste trabalho, a ofuscação de atributos é bastante interessante para manter a sutileza das anonimizações realizadas no vídeo. Para tal, é necessário desenvolver, ou analisar e avaliar as soluções já existentes, e adaptar este trabalho para ofuscar os atributos, no lugar do filtro de embaçamento.

7.2.4. Anonimização de crianças e bebês

Pessoas abaixo da maioridade penal podem ser alvos para um trabalho de anonimização automática de pessoas. Seria interessante um trabalho que generalizasse este projeto, inserindo a possibilidade de anonimizar pessoas baseadas em suas idades. A grande dificuldade, aos olhos do autor deste trabalho, está em evitar falso positivos ou negativos, uma vez que existe um alto grau de sutileza em relação à aparência e idade real.

7.2.5. Avaliar alternativas à MLP

Um MLP não é a única maneira possível de classificar pessoas utilizando os *embeddings* gerados. Um outro método menos inteligente, por assim dizer, é utilizar similaridade por cosseno[17], que consiste em comparar dois vetores utilizando a função trigonométrica

[17] `https://pt.wikipedia.org/wiki/Similaridade_por_cosseno`

cosseno. Por outro lado, também existem outras formas de utilizar MLPs, como proposto por [Capello et al. 2009], que propõe a utilização de, ao invés de um único MLP com múltiplas classes, um MLP por classe.

Referências

Barbu, T. (2010). Gabor filter-based face recognition technique. *Proceedings of the Romanian Academy*, 11(3):277–283.

Botalb, A., Moinuddin, M., Al-Saggaf, U., and Ali, S. S. (2018). Contrasting convolutional neural network (cnn) with multi-layer perceptron (mlp) for big data analysis. In *2018 International conference on intelligent and advanced system (ICIAS)*, pages 1–5. IEEE.

Candido, J. P. S., de Araújo, T. F., and Ribeiro, W. A. C. (2021). Histórico da lei geral de proteção de dados (lgpd).

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Capello, D., Martínez, C. E., Milone, D. H., and Stegmayer, G. (2009). Array of multilayer perceptrons with no-class resampling training for face recognition.

Deng, J., Guo, J., Ververas, E., Kotsia, I., and Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5203–5212.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Desai, M. and Shah, M. (2021). An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (mlp) and convolutional neural network (cnn). *Clinical eHealth*, 4:1–11.

Gardner, M. W. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636.

Grosselfinger, A.-K., Münch, D., and Arens, M. (2019). An architecture for automatic multimodal video data anonymization to ensure data protection. In Bouma, H., Prabhu, R., Stokes, R. J., and Yitzhaky, Y., editors, *Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies III*, volume 11166, pages 206 – 217. International Society for Optics and Photonics, SPIE.

Jadhav, A., Lone, S., Matey, S., Madamwar, T., and Jakhete, S. (2021). Survey on face detection algorithms. *International Journal of Innovative Science and Research Technology*, 6(2).

Kortli, Y., Jridi, M., Al Falou, A., and Atri, M. (2020). Face recognition systems: A survey. *Sensors*, 20(2):342.

Kumar, A., Kaur, A., and Kumar, M. (2019). Face detection techniques: a review. *Artificial Intelligence Review*, 52(2):927–948.

Li, J., Han, L., Chen, R., Zhang, H., Han, B., Wang, L., and Cao, X. (2021). Identity-preserving face anonymization via adaptively facial attributes obfuscation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3891–3899.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220.

Pulfer, E.-M. (2019). Different approaches to blurring digital images and their effect on facial detection.

Qi, D., Tan, W., Yao, Q., and Liu, J. (2021). Yolo5face: why reinventing a face detector. *arXiv preprint arXiv:2105.12931*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shams, B., Nasim, F. I., et al. (2019). *Analysis on face recognition based on five different viewpoint of face images using MTCNN and FaceNet*. PhD thesis, Brac University.

Sharif, M., Naz, F., Yasmin, M., Shahid, M. A., and Rehman, A. (2017). Face recognition: A survey. *Journal of Engineering Science & Technology Review*, 10(2).

Smelyakov, K., Chupryna, A., Bohomolov, O., and Hunko, N. (2021). The neural network models effectiveness for face detection and face recognition. In *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pages 1–7.

Tran, P. (2021). Fast video-based face recognition in collaborative learning environments. *arXiv preprint arXiv:2110.14720*.

Turk, M. (2005). Eigenfaces and beyond. *Face Processing: Advanced Modeling and Methods*, pages 55–86.

Yang, S., Luo, P., Loy, C. C., and Tang, X. (2016). Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, J. and Pun, C.-M. (2020). Personal privacy protection via irrelevant faces tracking and pixelation in video live streaming. *IEEE Transactions on Information Forensics and Security*, 16:1088–1103.