UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

LUCAS DE BRITO SILVA

PROCESSAMENTO DIGITAL DE IMAGENS

PROF. APARECIDO NILCEU MARANA

RECONHECIMENTO BIOMÉTRICO FACIAL E OCULAR

BAURU

2020

Sumário

Dataset

O *dataset* escolhido para a realização do trabalho foi o ARFACE (MARTINEZ. Aleix M, 1998), o qual, originalmente, foi desenvolvido na universidade de Ohio, contando com a participação de 126 pessoas (70 homens e 56 mulheres) e tendo mais de 4000 imagens em diferentes poses, expressões faciais, condições de iluminação e uso de acessórios, segundo consta no site oficial do *dataset* (http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html <Acesso em nov. 2020>). Todavia, para esse trabalho, foram utilizadas apenas 3257 imagens da face toda, compreendendo 136 pessoas e 26 figuras diferentes de cada uma delas. Supõe-se que o *dataset* contou com novas imagens no decorrer dos anos.

A descrição das 26 categorias de imagens se dá por:

1. Expressão neutra;
2. Sorrindo;
3. Raiva;
4. Gritando;
5. Luz ao lado direito;
6. Luz ao lado esquerdo;
7. Todas as luzes acesas;
8. Usando óculos escuros;
9. Com óculos de sol e luz ao lado direito;
10. Com óculos de sol e luz ao lado esquerdo;
11. Usando cachecol;
12. Usando cachecol e luz à direita;
13. Com cachecol e luz à esquerda;

14 à 26. Segunda sessão (mesmas condições que 1 à 13).

MTCNN

Para realizar a detecção facial, assim como os pontos do nariz, olho direito e olho esquerdo e extremidades da boca, foram realizadas com o auxílio do framework *Multi-task Cascaded Convolutional Networks* (MTCNN), baseada no trabalho realizado por ZHANG. Kaipeng et al., 2016, sendo esta uma desenvolvida como uma solução para a detecção e o alinhamento da face. Englobando três estágios de redes convolucionais que são capazes de reconhecer rostos e pontos de referência supracitados.

Nesse projeto, a sua implementação foi feita com o auxílio de uma biblioteca já treinada, em Python, que recorre às tecnologias Tensorflow e Keras. A biblioteca citada encontra-se disponível em: https://pypi.org/project/mtcnn/ <Acesso em nov. 2020>.

Para questões de aferição, a biblioteca disponibiliza uma tabela (tabela 1) com desempenhos testados em imagens individuais e com um computador Core i7, a qual pode ser consultada abaixo.

Tabela 1: desempenho do MTCNN com uma única imagem.

Tamanho da imagem	Total de pixels	Tempo de processamento	FPS
460x259	119.140	0,118 segundos	8.5
561x561	314.721	0,227 segundos	4.5
667x1000	667.000	0,456 segundos	2.2
1920x1200	2.304.000	1,093 segundos	0.9
4799x3599	17.271.601	8,798 segundos	0.1

Quanto aos resultados obtidos a partir das detecções no *dataset* com 3257 imagens, utilizando um computador Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz, observou-se que o processo, levou por 189 minutos (3h15min), sendo dividido em três dias, visto que pelo excesso de requisições, em determinado momento as detecções paravam. A maior parte das imagens detectadas para as imagens masculinas foram as imagens de *labels* 2, 3, 4 e 5, ou seja, fotos sorrindo, com raiva, gritando e com luz ao lado direito, tendo uma detecção total de 76 imagens para ambas as classes. Abaixo, segue um exemplo das classes supracitadas (figura 1).

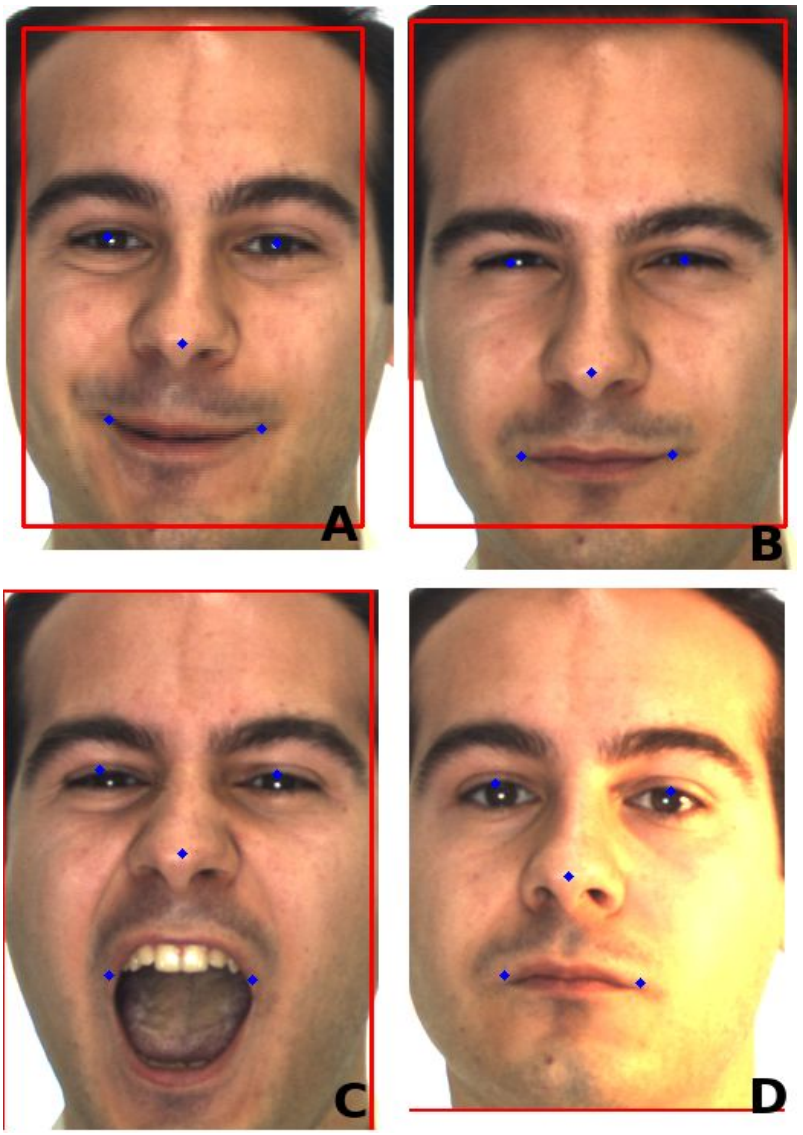

Figura 1: imagem de *label* 2(A), 3(B), 4(C) e 5(D) (do autor).

Para as imagens femininas, a *label* mais detectada foi a 2, ou seja, de pessoas sorrindo. A qual é exemplificada na figura 2:

Figura 2: imagem de *label*=2 (do autor).

Por outro lado, visualiza-se que a rede MTCNN teve dificuldades para detectar imagens de *label* 20, 7 e 26 para os homens, detectando apenas 44, 46 e 49 imagens, respectivamente, em que as classes são exemplificadas a seguir pela figura 3.

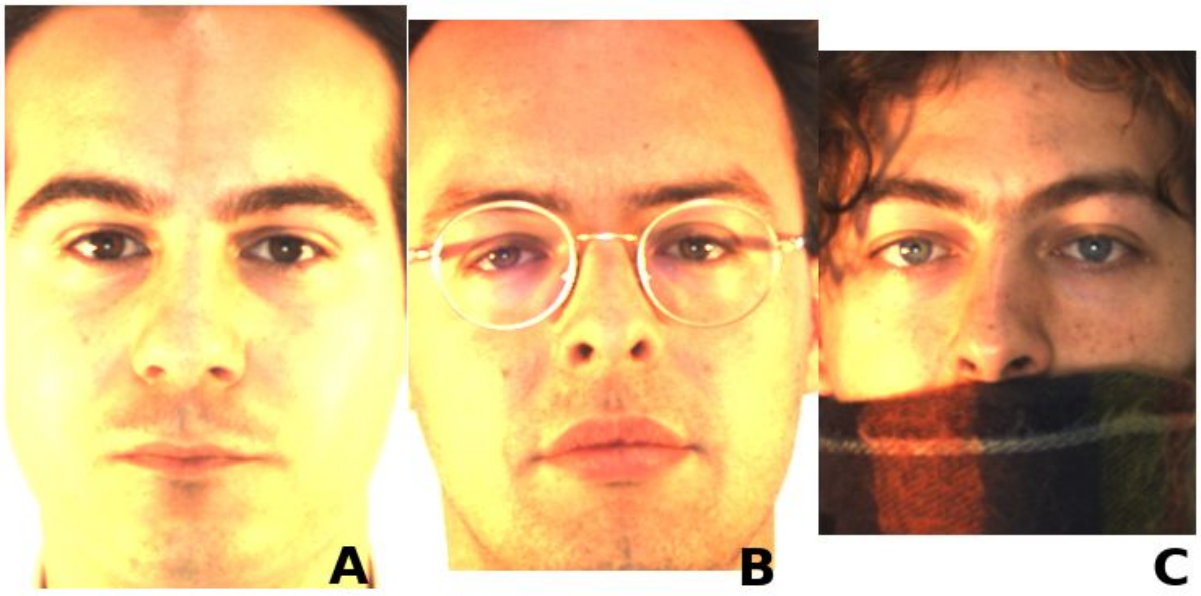

Figura 3: imagem de *label* 20 (A), 7(B) e 26(C) (MARTINEZ, Aleix M. 1998).

Para as imagens femininas os *labels* pelos quais a rede teve dificuldade foram os mesmos, tendo apenas uma diferença quanto à ordem, visto que as imagens de *labels* menos detectadas foram 26, com 26 fotos e os *labels* 7 e 20, tendo ambas 31 figuras dessa classe. A representação encontra-se na imagem abaixo (figura 4):

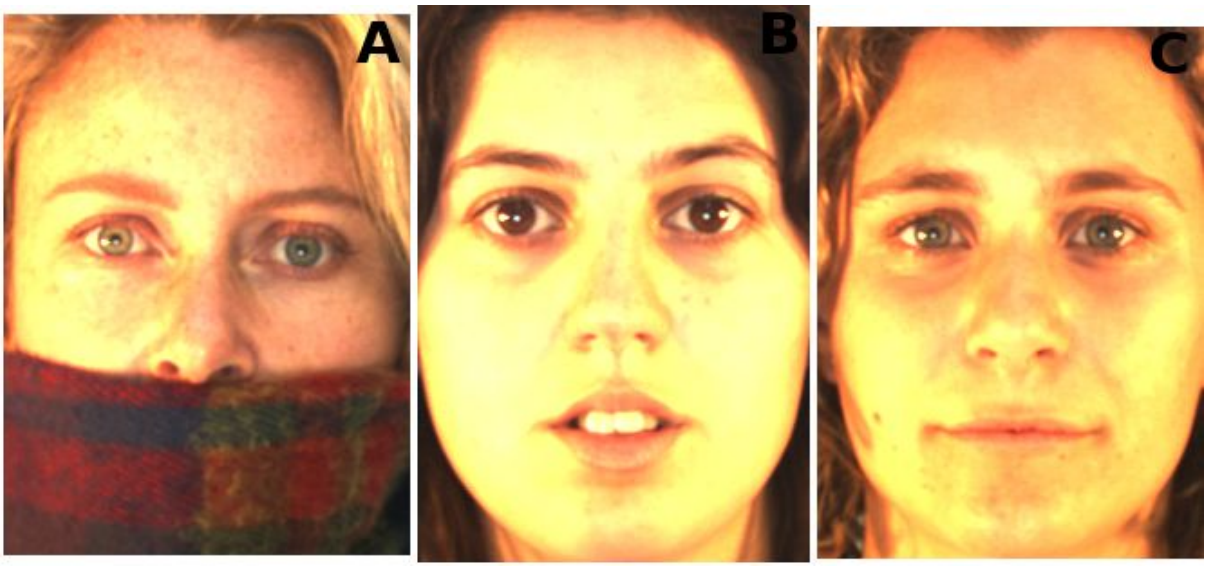

Figura 4: imagem de *label* 26 (A), 7(B) e 20(C) (MARTINEZ, Aleix M. 1998).

Vale ressaltar que as imagens de classe 7 e 20 possuem as mesmas condições, com uma diferença apenas nos dias.

De modo geral, dentre as 3257 imagens de face, a rede conseguiu detectar apenas 2955 faces, sendo que essas foram utilizadas para os testes posteriores com um tamanho padronizado de 224 x 224 px.

Descritores faciais

Local Binary Pattern (LBP)

Padrão binário local, ou em inglês, local binary pattern (LBP) é o nome que se dá a técnica desenvolvida por OJALA, T., 2002, que consiste em computar e representar localmente uma área pela sua textura e essa representação é devida ao valor de um determinado pixel e seus vizinhos.

De modo sucinto, o processo do LBP consiste em transformar uma imagem em escala de cinza e selecionar os pixels de uma determinada vizinhança a partir do pixel central. Tendo feito isso, o pixel central passa a ser um limiar, em que os vizinhos maiores ou iguais que o mesmo são transformados em 0 e os menores são transformados em 1, o que é retratado na figura abaixo (figura 5), considerando uma janela 3x3.

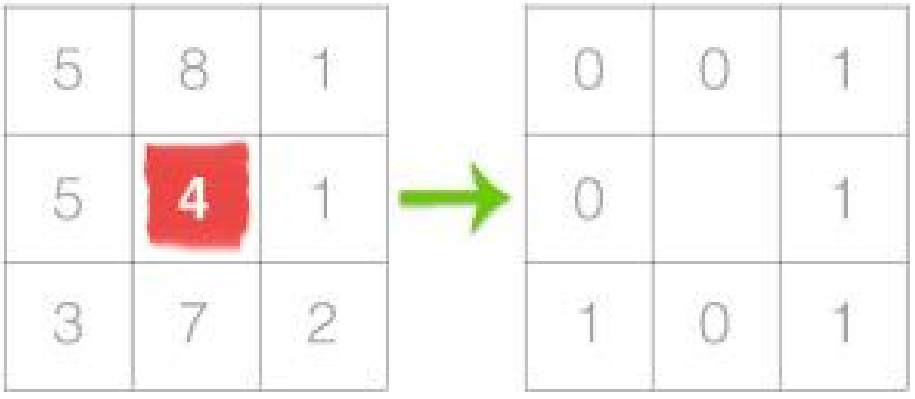

Figura 5: exemplo de uma transformação LBP com kernel 3x3. Disponível em: <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>. Acesso em nov. 2020.

Tendo em vista que em um kernel 3x3 tem-se 8 vizinhos em relação ao pixel central, pode-se dizer que se tem 2^8= 256 possibilidades de para padrões de binários. Após o passo supracitado, um dos pixels vizinhos deve ser selecionado para iniciar o cálculo do local, salvo que a posição desse pixel, assim como o sentido (horário ou anti-horário), devem ser mantidos para todo o cálculo de LBP da imagem. Após isso, os resultados dessa seleção binária são armazenados em um

vetor de 8 bits e posteriormente para decimal quando com o bit 1, assim como na figura 6.

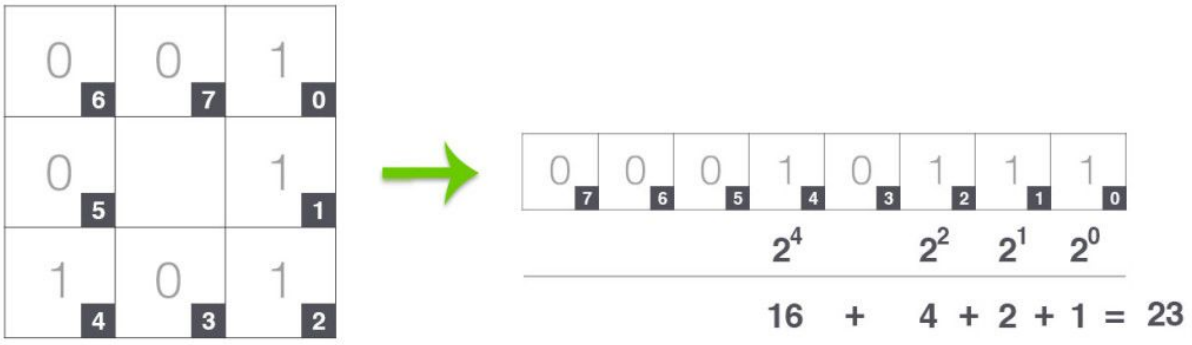

Figura 6: exemplo de uma transformação de binário para decimal. Disponível em: <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>. Acesso em nov. 2020.

A saída desse processo resultará em uma imagem composta desses números decimais e o último dos passos é computar um histograma de modo que este contenha as características da imagem, ou melhor, os descritores faciais. Um exemplo é demonstrado na imagem abaixo (figura 7).

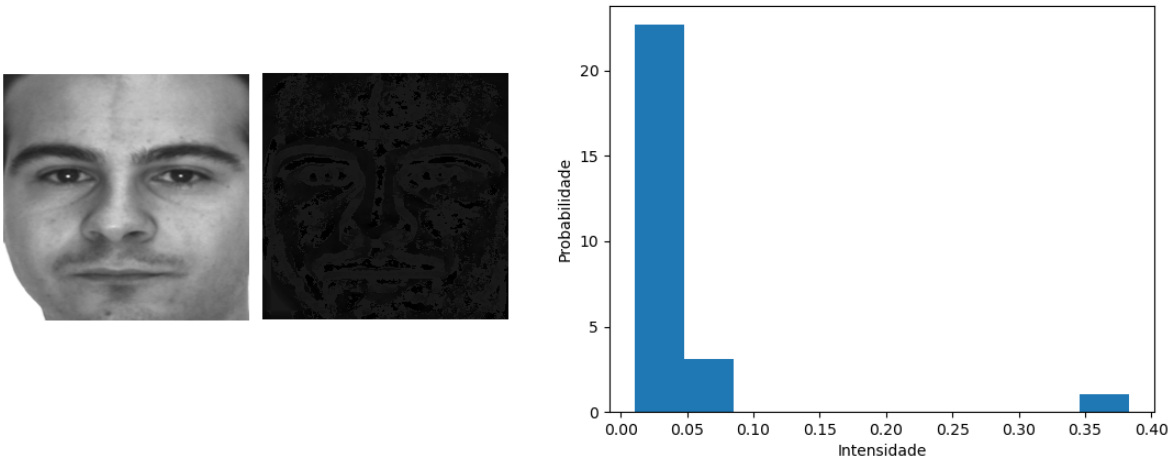

Figura 7: imagem original, imagem em LBP e histograma, respectivamente (do autor).

O desenvolvimento ocorreu com o auxílio da biblioteca scikit-learn que proporciona como saída do LBP um histograma como o que é necessário para a determinação de um padrão de descritores faciais assim como é invariante à rotação e escala de cinza.

Nos testes desenvolvidos o *dataset* das imagens que foram identificadas pela MTCNN foi dividido em duas partes, sendo 80% para treino e 20% para teste. Após essa divisão, o algoritmo utilizado para classificação foi a Máquina de Vetores de Suporte, do inglês Support Vector Machine (SVM), que obteve em sua execução, na divisão de teste com 296 imagens, 77 acertos e 222 erros. Ou seja, obteve 25% de

acerto no *dataset*, considerando que foram utilizados 40 pontos em uma vizinhança de 8 pixels.

VGGFace

O uso de *deep learning* tem crescido cada vez mais quando se trata de identificação e trabalho com imagens, dessa forma dentre os modelos desenvolvidos pela faculdade de Oxford que é considerado estado da arte para a detecção e identificação facial, cita-se VGGFace e VGGFace2, as quais foram fruto de outras redes do grupo Visual Geometry Group (VGG).

A denominada VGGFace foi publicada por Parkhi. 2015, em que o nome original da publicação era *Deep Face Recognition*. Sua motivação era poder ser comparada com redes como as do Google a Facebook, então realizou seu treinamento com mais de dois milhões de faces, utilizando uma função de ativação *softmax* na camada de saída para classificar as faces como pessoas. Esta camada é então removida para que a saída da rede seja uma representação vetorial da face.

Logo depois, em 2017, Qiong Cao, et al. escreveu um *paper* denominado *VGGFace2: A dataset for recognizing faces across pose and age*, em que sua intenção era ter um *dataset* para a para treinar e avaliar modelos de reconhecimento facial mais eficazes. O *dataset* continha 3,31 milhões de imagens de 9131 indivíduos, com uma média de 362,6 imagens para cada indivíduo. Os modelos são treinados no conjunto de dados, especificamente um modelo ResNet-50 e um modelo SqueezeNet-ResNet-50 (chamado SE-ResNet-50 ou SENet).

Quanto ao desenvolvimento do trabalho, foi utilizado da técnica de *transfer learning* no modelo original da VGGFace2 com a resnet50, os quais estão disponíveis em: https://github.com/rcmalli/keras-VGGFace <Acesso em dev. 2020>. Para realizar a técnica de *transfer learning* a última camada da rede foi retirada e substituída por algumas camadas, sendo elas GlobalAveragePooling2D, em que o *output* de todas as celebridades utilizadas para essa rede (8631 celebridades) passa a ser resumida em uma média, mantendo a profundidade. O que é exemplificado na imagem abaixo (figura 8):

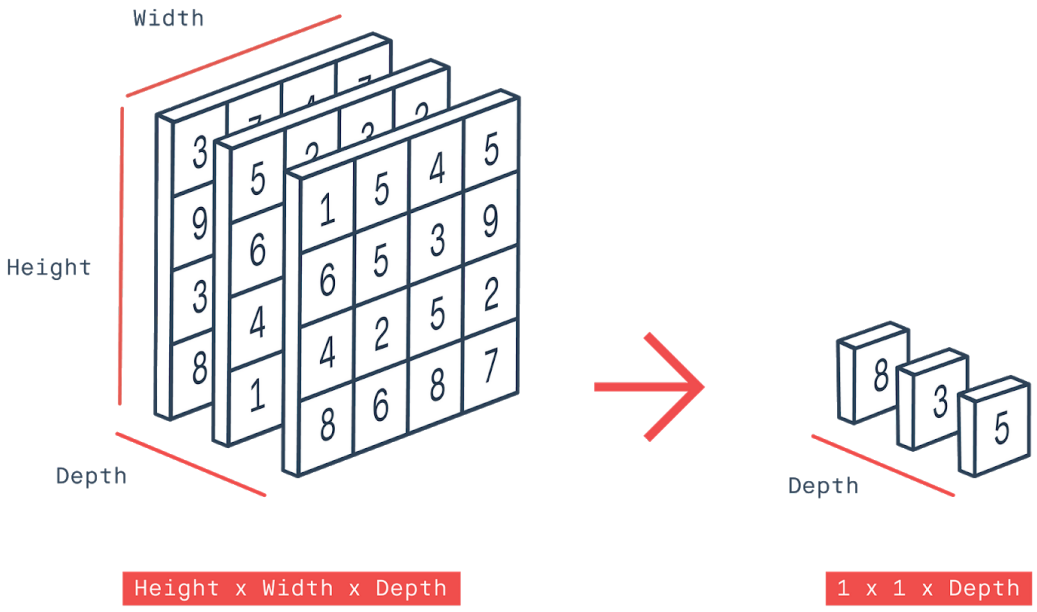

Figura 8: exemplo de GlobalAveragePooling2D. Disponível em: <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-global-average-pooling>. Acesso em dez. 2020.

Após esse processo, foram adicionadas duas camadas densas, assim como a imagem abaixo (figura 9), que representam neurônios totalmente conectados. Na primeira dessas camadas, encontram-se 256 nós e 199 na segunda, os quais foram escolhidos empiricamente.

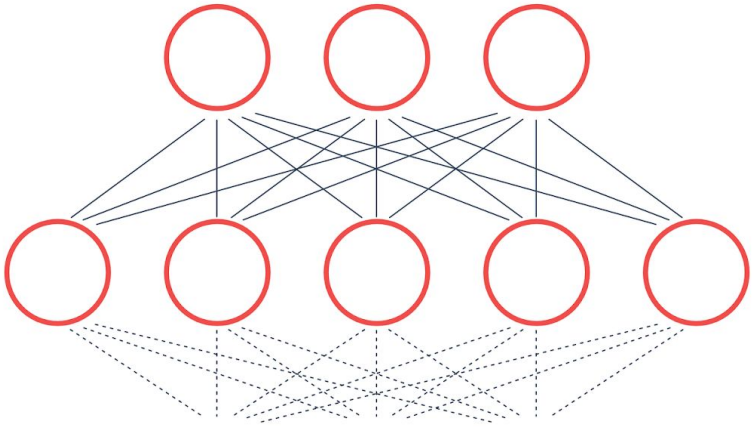

Figura 9: exemplo de camadas densas. Disponível em: <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/dense>. Acesso em dez. 2020.

Por fim, antes da última camada densa que tem a mesma quantidade de nós que a de classes (135, considerando os homens e as mulheres), o qual exibirá a

probabilidade de cada classe, tem-se um bloco de *dropout*. Esse bloco é responsável por eliminar alguns nós de neurônios (aleatoriamente), de modo que o modelo fique menos sujeito à supertreinamento, do inglês, *overfitting*. No projeto, foram eliminados 20% dos neurônios, nessa fase. A figura 10 exemplifica o que foi descrito acima.

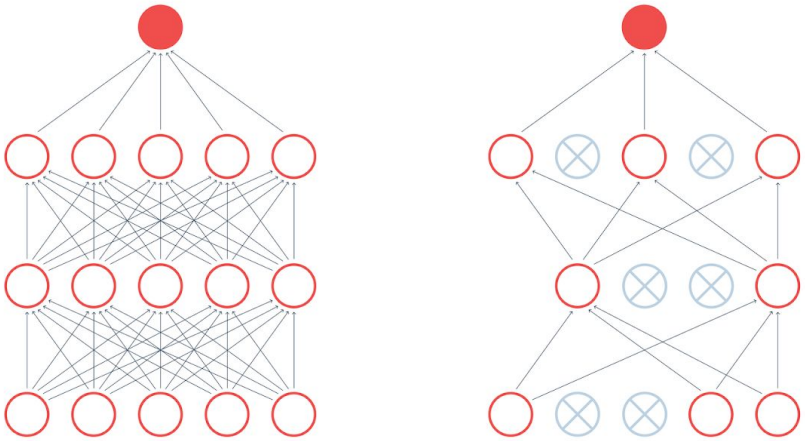

Figura 10: exemplo de *dropout*. Disponível em: <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/dropout>. Acesso em dez. 2020.

As últimas camadas, que englobam as descritas acima podem ser visualizadas na figura 11, assim como o modelo na totalidade está disponível em: https://drive.google.com/file/d/1GaEKoU4_8TMCnj9NfsQAX1eLplpfrOho/view?usp=sharing.

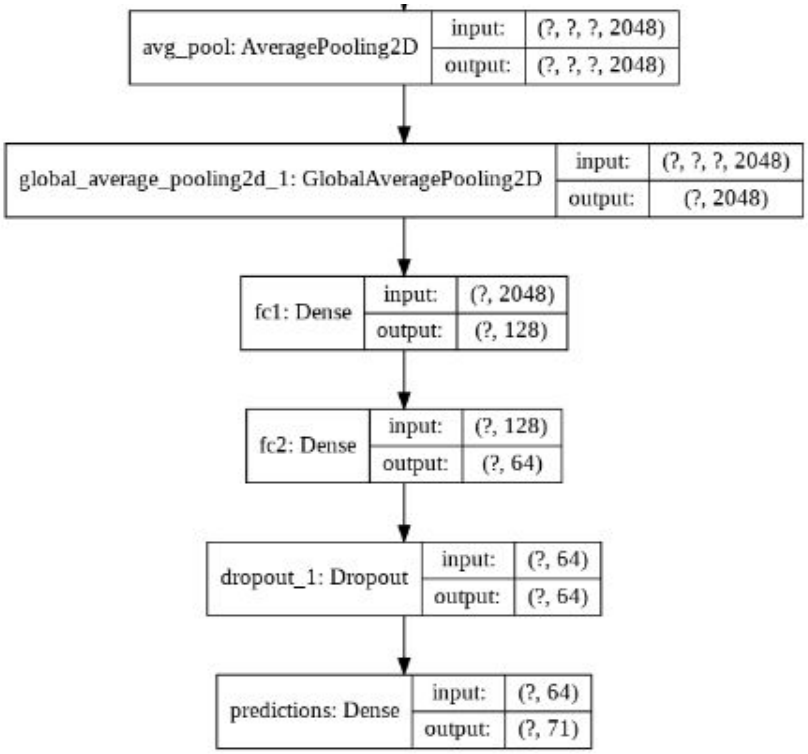

Figura 11: final do modelo VGGFace2. (do autor).

Quanto aos acertos e erros da rede, pode-se notar que o seu desempenho foi baixo comparado com o método *handcrafted* LBP, em que na base de teste com 296 imagens, o modelo esteve acertando 5 das classes e errando 291, o que representa 1,68% de acertos. Vale destacar que o modelo foi treinado utilizando apenas 5 épocas, mantendo suas camadas congeladas e para obter um *fine-tunning*, mais hipóteses deveriam ser testadas como o aumento de camadas densas, descongelamento das camadas e maior variedade de épocas, especulações as quais não foram testadas por conta do tempo e limitação de equipamento físico.

Descritores oculares

Tendo em vista os métodos já explicados anteriormente, os seguintes testes consistiam na replicação da aplicação do método da VGGFace e LBP para as regiões oculares, de modo que fosse possível encontrar um avanço quanto aos indicadores de assertividade.

Vale destacar que as configurações foram as mesmas dos modelos citados anteriormente, todavia com a ausência de participação da MTCNN, visto que é uma rede específica para detecção facial, logo não encontraria nada por não ter todos os elementos de um rosto.

Além disso, a divisão do *dataset* para treino e teste, mantiveram-se em 80/20, porém o dataset do lado direito e do lado esquerdo estiveram juntos, resultando assim em um *dataset* com uma quantidade maior de imagens.

Local Binary Pattern (LBP)

O método testado consistiu em uma configuração de vizinhança 8, tendo 40 pontos distribuídos entre a vizinhança, o que resulta em 5 pontos por pixel, considerando uma vizinhança circular, semelhante à figura 12. Para realizar o treinamento, foram utilizadas 5211 imagens, sendo que no teste, o modelo com o uso de SVM acertou 135 pessoas e errou 517 pessoas, a partir de sua região ocular direita ou esquerda. Considerando que o dataset de teste era composto por 652 amostras, pode-se dizer que o modelo obteve 20,7% de acerto, colocando-o abaixo da detecção pela face inteira.

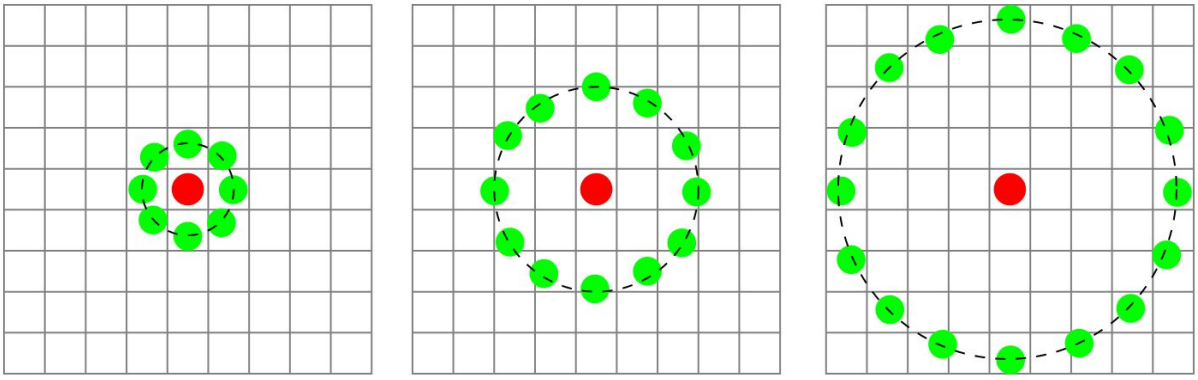

Figura 12: exemplo de distribuição de pontos em vizinhança. Disponível em: <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>. Acesso em nov. 2020.

VGGFace

Já quanto ao método VGGFace, tornou-se notória a diferença de tempo para o treinamento, em que devido ao maior número de amostras (5211 amostras) o treinamento foi mais demorado (por volta de 50 minutos no Google Collaboratory), mas se tratando dos resultados, a VGGFace obteve 0,61% de acerto, visto que dentre as 652 imagens, acertou 4 e errou 648.

Vale a pena lembrar que o modelo utilizado teve as mesmas configurações do que o modelo para detecção da face como um todo, assim, evidenciando que se fosse aplicado uma arquitetura de rede melhor e com mais hipóteses, provavelmente o modelo teria uma assertividade maior.

Comparado ao modelo VGGFace testado na face completa, o modelo utilizando apenas as regiões oculares se saiu pior, tendo em vista que houve uma diferença de 1,06% entre as porcentagens, desconsiderando o tamanho do *dataset*.

Métricas de desempenho

As métricas de desempenho, muito mais que mostrar os acertos e erros, mostram o desempenho do modelo utilizado, tendo que esse pode ser avaliado por diversos parâmetros.

Nessa seção alguns deles serão citados, de modo que seja possível, também, ter um comparativo entre as redes desenvolvidas até o presente momento.

Precisão - Revocação

A Precisão-Revocação é uma medida normalmente utilizada para previsão de quando as classes estão muito desequilibradas. Na recuperação de informações, a precisão é uma medida da relevância dos resultados, enquanto a revocação é uma medida de quantos resultados verdadeiramente relevantes são devolvidos.

Em sua curva é possível visualizar a troca entre precisão e revocação para diferentes limiares. Uma área alta sob a curva representa tanto a alta revocação (baixa taxa de falsos negativos), quanto a alta precisão (baixa taxa de falsos positivos). Pontuações altas, no geral, demonstra resultados precisos, bem como sendo normalmente resultados positivos.

A precisão (P) é definida como o número de verdadeiros positivos (V_p) sobre o número de verdadeiros positivos mais o número de falsos positivos (F_p).

$$P = \frac{V_p}{V_p + F_p} \tag{1}$$

Já a revocação (R) é definida como o número de verdadeiros positivos (V_p) sobre o número de verdadeiros positivos mais o número de falsos negativos (F_p).

$$P = \frac{V_p}{V_p + F_n} \tag{2}$$

Quanto aos modelo LBP, as curvas obtidas com a métrica de precisão - revocação para as imagens faciais e de região oculares foram as seguintes:

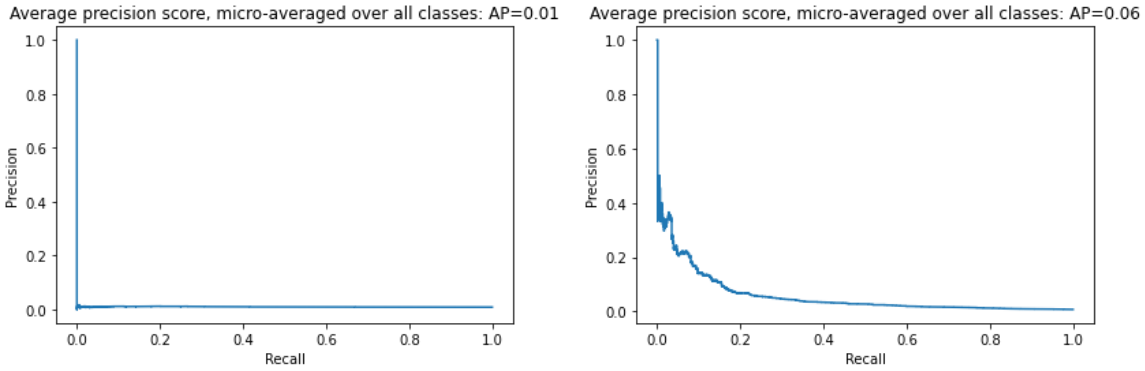

Figura 13: curva de precisão - revocação das imagens faciais e das regiões oculares do modelo LBP, respectivamente. (do autor).

Já o modelo VGGFace2 obteve as seguintes curvas em relação à métrica de precisão - revocação:

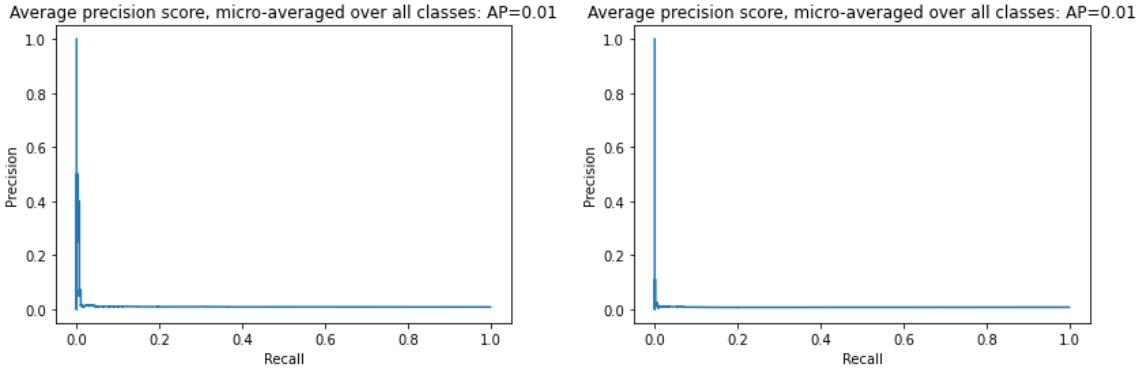

Figura 14: curva de precisão - revocação das imagens faciais e das regiões oculares do modelo VGGFace2, respectivamente. (do autor).

Observando as curvas é possível concluir que não foram obtidas boas precisões. Todavia, dentre os resultados de LBP, destaca-se o modelo das regiões oculares, diferente do modelo da VGGFace2 que possui uma igualdade de *average precision*, em que temos um destaque no modelo com a face como um todo na parte inicial.

F-measure

O F-measure, ou F-1 (como também é conhecido), é o resultado que faz uso das medidas descritas à priori (precisão e revocação), visto que sua definição ($F1$) é fruto de uma média ponderada da precisão(P) e recall, onde uma pontuação $F1$ atinge seu melhor valor em 1 e a pior pontuação em 0. Sua fórmula pode ser descrita por:

$$F1 = 2\frac{P * R}{P + R} \quad (3)$$

Quanto aos resultados do projeto, o F-measure para LBP facial e ocular, foram, respectivamente, iguais à 0.25 e 0.167, sendo que para os modelos de VGG Face foram iguais à 0.01 e 0.001.

AUC e ROC

Do inglês, Area Under the Curve (AUC) ou Area Under the Receiver Operating Characteristic (AUROC) é uma métrica muito utilizada para problemas de classificação que conta com vários ajustes de limiares. ROC é uma curva de probabilidade e a AUC representa o grau ou medida de separabilidade, sendo que essa métrica informa o quanto o modelo é capaz de distinguir entre as classes. Quanto maior a AUC, melhor o modelo está prevendo as classes corretamente. Quanto mais bem separada forem os falsos e positivos , melhor será a curva ROC.

A curva ROC é traçada com TPR (True Positive Receiver) contra a FPR (False Positive Receiver) onde TPR está no eixo y, e FPR está no eixo x. Todavia, vale destacar que em algumas documentações esses parâmetros podem variar.

Outro ponto a se destacar é que a curva ROC conta com um limiar que varia entre o TPR e o FPR, sendo que esta variação pode ser de acordo com o problema a ser solucionado. Um exemplo disso são os sistemas de segurança, que contam com um FAR baixo e limiar alto. Em contrapartida, em sistemas forense o FAR pode ser alto, tendo um limiar mais baixo.

Para exemplificar o comportamento de ROC e AUC, sendo ROC é uma curva de probabilidade, serão traçadas distribuições de probabilidades, tendo a curva de distribuição vermelha como classe positiva e a curva de distribuição verde como classe negativa.

Uma situação ideal ocorre quando duas curvas não se sobrepõem, ou seja, o modelo tem uma medida ideal de separação, sendo perfeitamente capaz de distinguir as classes.

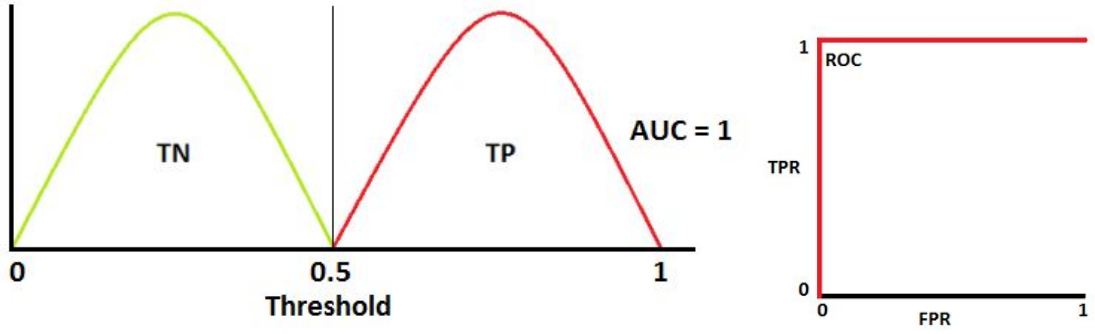

Figura 15: AUC e ROC ideais. Disponível em: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. Acesso em dez. 2020.

Quando duas distribuições se sobrepõem, tem-se dois erros que podem ser minimizados ou maximizados, dependendo do limiar adotado. Quando a AUC é de 0.7, por exemplo, há 70% de chance de que o modelo seja capaz de distinguir entre classe positiva e classe negativa.

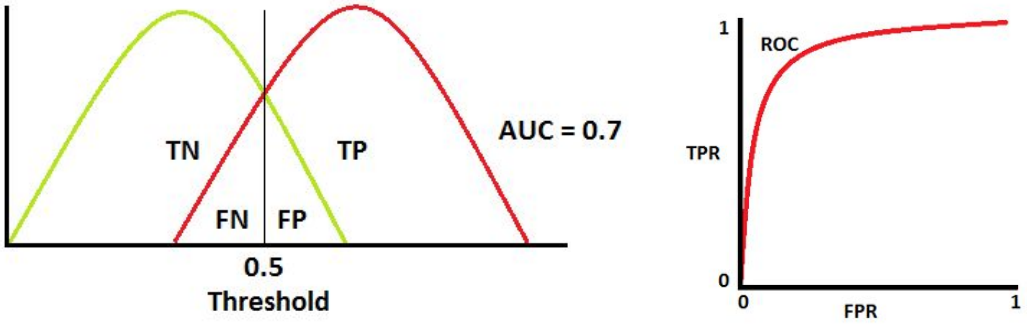

Figura 16: AUC e ROC com 70% de distinção. Disponível em: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. Acesso em dez. 2020.

Uma situação ruim se dá quando a AUC é de aproximadamente 0.5, em que o modelo não tem capacidade de discriminação para distinguir entre classe positiva e classe negativa.

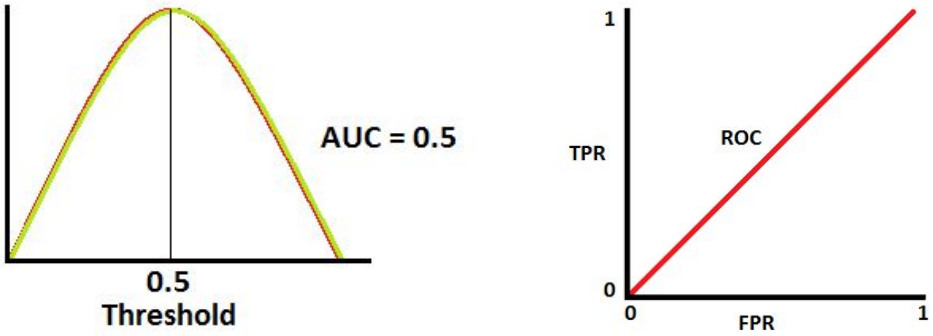

Figura 17: AUC e ROC sem distinção. Disponível em: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. Acesso em dez. 2020.

A última situação ocorre quando a AUC é aproximadamente 0, o modelo está, na verdade, trocando as classes, onde este prevê a classe negativa como uma classe positiva e vice versa.

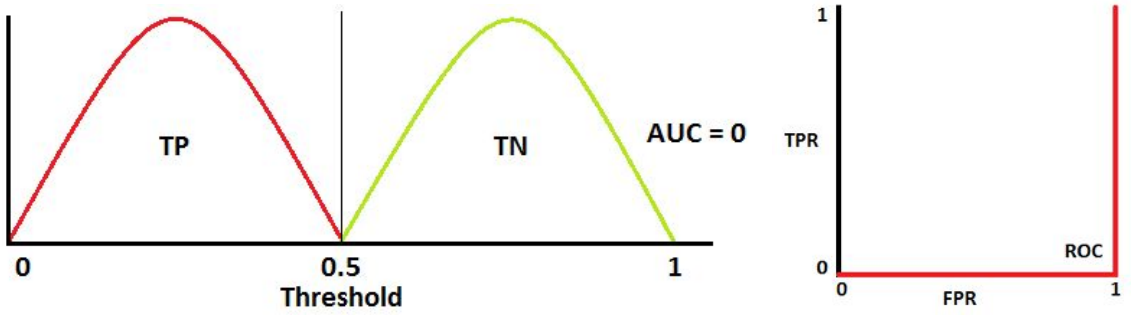

Figura 18: AUC e ROC trocando classes. Disponível em: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. Acesso em dez. 2020.

No projeto desenvolvido, as curvas obtidas são demonstradas a seguir, sendo que a primeira delas (LBP Facial) e segunda (VGGFace Facial) encontram-se mais próximas do caso em que as curvas trocam de classe por esta com uma área próxima de 50%.

A terceira, por sua vez, também foi muito próxima das anteriores, todavia esta teve destaque ao obter um valor de área maior que as demais (0,67) com um valor mais próximo ao eixo inicialmente e um afastamento a posteriori.

Por fim, a última das curvas demonstra um resultado intermediário entre a primeira e segunda curva, porém de modo geral, todas demonstram um relacionamento próximo de 50% em que as classes são sem distinção consistente.

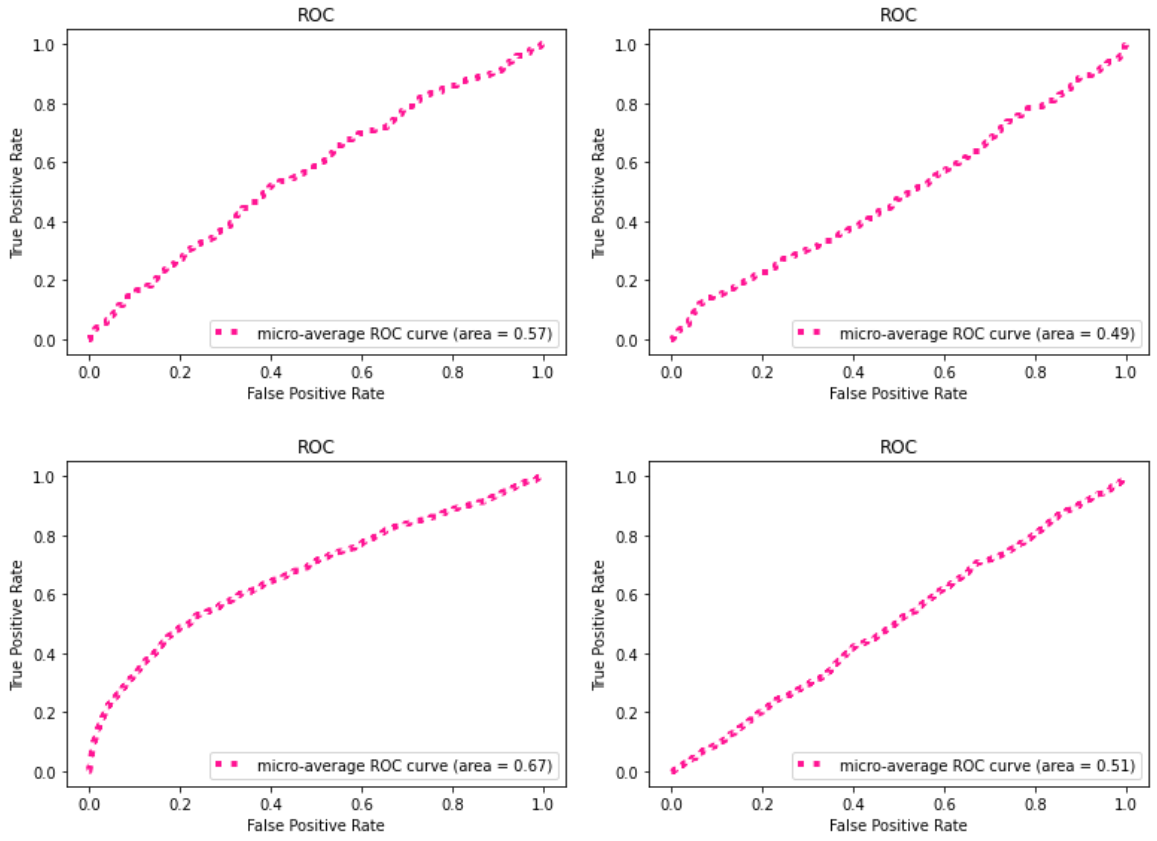

Figura 19: AUC e ROC do modelo em LBP facial, VGGFace facial, LBP ocular, e VGGFace ocular, respectivamente (da esquerda para direita, de cima para baixo). (do autor).

CMC

Uma curva CMC é usada para avaliar a precisão dos algoritmos que produzem uma lista ordenada de possíveis combinações entre valor provável e porcentagem. No caso do reconhecimento facial, a saída do algoritmo (VGGFace, por exemplo) seria uma lista de rostos, sendo que o rosto genuíno carregaria consigo a maior probabilidade.

Em geral, quanto melhor for o algoritmo, maior será a porcentagem CMC de classificação.

Em relação ao trabalho desenvolvido, foi feito um método que utilizou de uma classe CMC, a qual era responsável por formar e plotar a curva CMC selecionando aleatoriamente 5 das classes entre o dataset, todavia, infelizmente as métricas foram muito baixas o que resultou em um gráfico vazio para o método LBP facial, mas tem um pequeno relevo por volta do rank 15 para a classe Cw0037, como se observa na figura 20, porém a exemplificação desse gráfico se faz na figura 21.

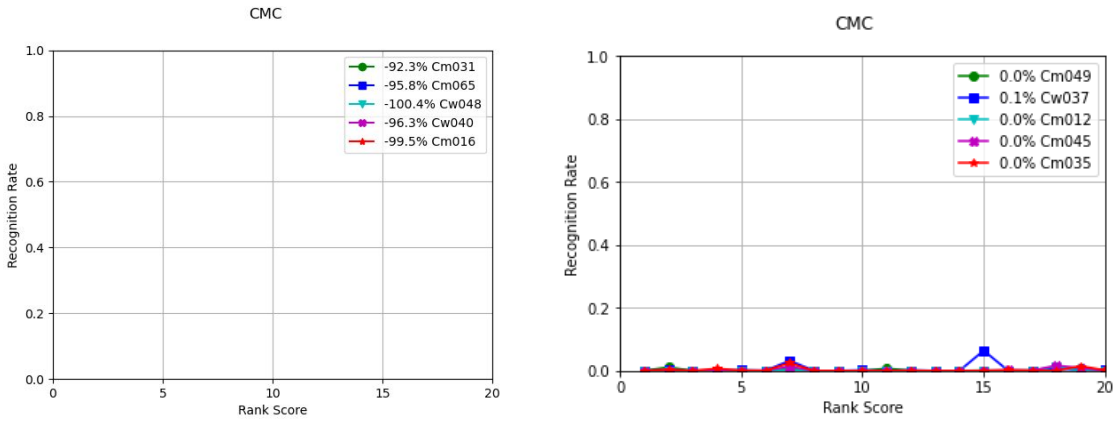

Figura 20: 5 classes do método LBP e VGGFace, em seguida, sendo representadas pelo gráfico de uma CMC. (do autor).

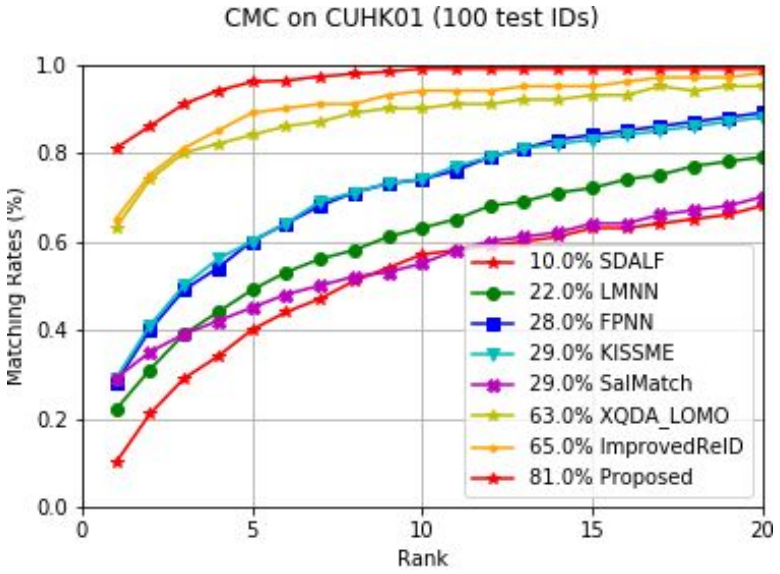

Figura 21: gráfico de representação de curvas CMC. Disponível em: <https://github.com/LinShanify/CMC>. Acesso em dez. 2020.

Equal Error Rate (EER)

Equal Error Rate (EER) é uma métrica que normalmente é utilizada em sistemas de segurança biométrica, de modo que predetermine os valores limiares para sua taxa de falsa aceitação (FAR) e sua taxa de falsa rejeição (FRR). Quando as taxas são iguais, o valor comum é referido como a ERR. Basicamente, seu valor indica qual proporção de FAR é igual à proporção de FRR. Quanto menor o valor da ERR, maior é a precisão do sistema biométrico.

O Equal Error Rate também pode ser chamado de taxa de erro cruzado (*crossover rate*) ou taxa de erro cruzado, do inglês *Crossover Error Rate* (CER).

A representação do EER faz-se presente na seguinte imagem:

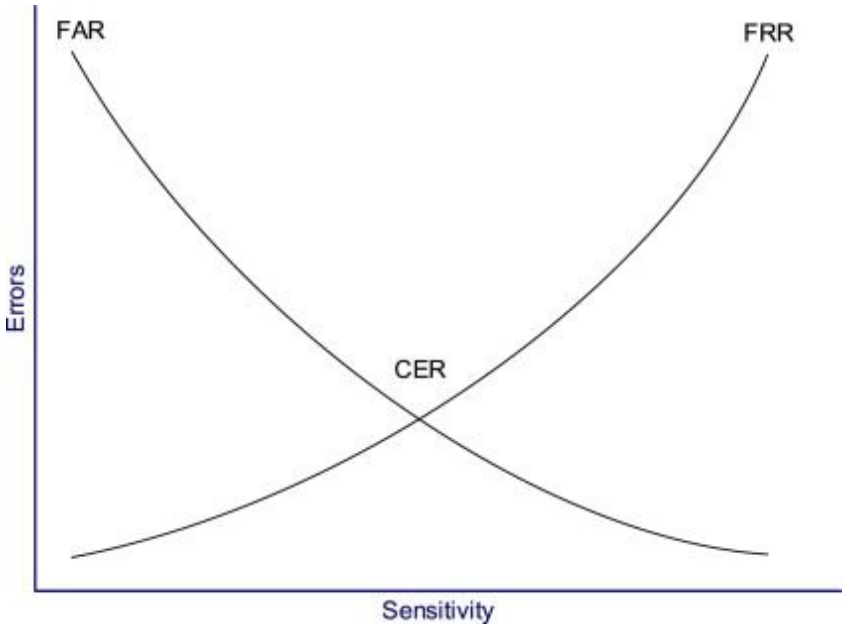

Figura 22: exemplo de EER. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978012811248900005X>. Acesso em dez. 2020.

No projeto, essa funcionalidade ocorreu em conjunto com o método de curva ROC, visto que lá já estavam disponíveis os valores de FPR e TPR, sendo assim os valores de ERR para o LBP facial, LBP Ocular, VGGFace facial e VGGFace ocular, foram, respectivamente:

- 0.452;
- 0.376;
- 0.511;
- 0.496.

Aplicação de filtros em faces neutras

Quanto aos testes com filtros, o objetivo final era concluir se com a mudança das imagens de face neutra, seria possível obter uma melhora quanto a assertividade dos métodos testados, todavia as faces neutras possuem apenas duas imagens por pessoas, pertencendo ao *label* 1 e 14, o que seria algo muito pequeno para treinamento dos métodos, assim como seus respectivos testes. Dessa forma, optou-se pela a aplicação de uma equalização de histograma em todas as imagens do *dataset*, de modo que fosse possível visualizar a influência dessa mudança em relação aos modelos.

Para que o processo não fosse repetido por inteiro, devido às limitações da MTCNN, as quais já foram citadas, o processo de detecção com a mesma foi descartado, assim, utilizando as imagens que já haviam sido detectadas por essa rede para aplicar a equalização de histograma.

Para realizar o processo de equalização do histograma da imagem, processo o qual distribui a luminosidade na imagem, foi utilizado a biblioteca OpenCV, que

possui uma função que aplica essa equalização. Porém, antes que a equalização fosse realizada, a imagem precisou passar por um processo de conversão de escala de cores, visto que se a aplicação fosse realizada diretamente na imagem original, que estava na escala RGB, as cores mudariam em vez de haver uma equalização de luminosidade. Dessa forma, a imagem original foi alterada para a escala de cores HSV, em que a camada valor (V), que representa a luminosidade, foi equalizada, e posteriormente houve uma nova mudança, em que a imagem passou para sua escala de cores original.

A diferença entre as duas imagens, com a aplicação de equalização e imagem original, podem ser visualizadas na figura 23.

Figura 23: imagem original e equalizada, respectivamente. (do autor).

Após a aplicação do método LBP foi possível observar as curvas de precisão - revocação (figura 24), todavia ambos não apontaram uma melhora significativa quanto ao reconhecimento, apresentando inclusive, uma piora quanto ao *average precision* com a região ocular. Em relação a VGGFace, os resultados de precisão - revocação podem ser observados na figura 25, em que temos algo muito similar às imagens que não receberam tratamento algum.

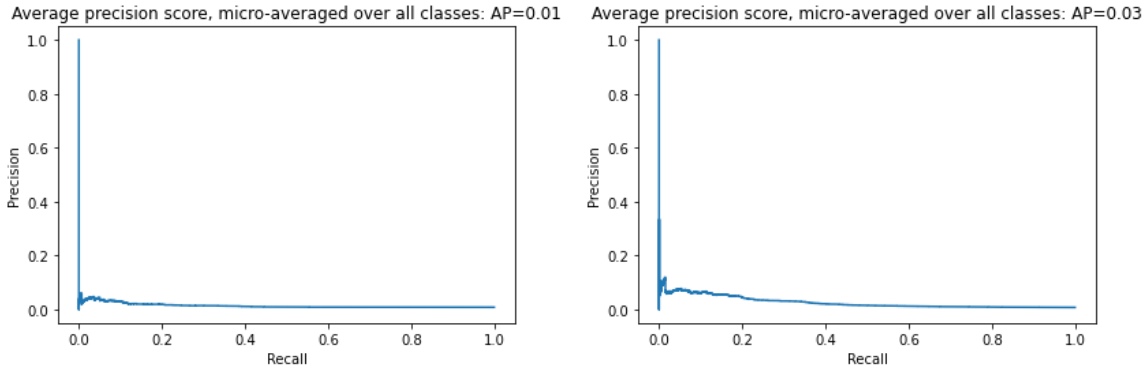

Figura 24:Precisão-Revocação dos métodos LBP facial e ocular equalizado, respectivamente. (do autor).

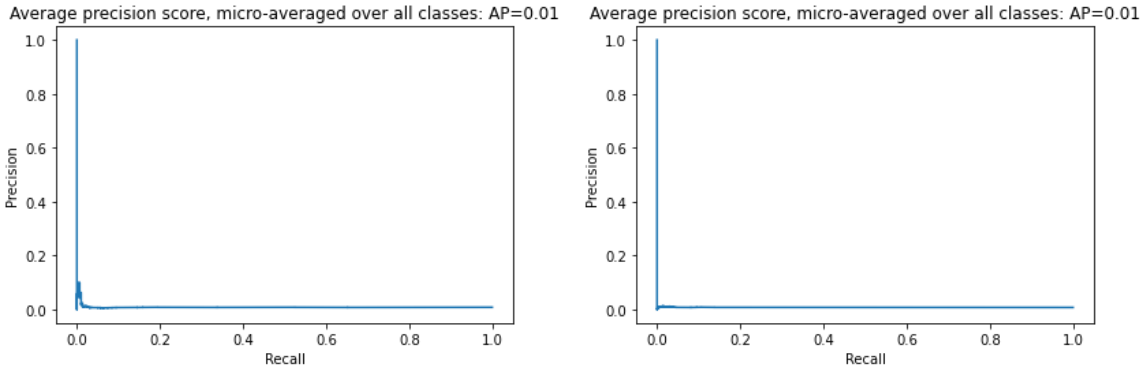

Figura 25:Precisão-Revocação dos métodos VGGFace facial e ocular equalizado, respectivamente. (do autor).

Quanto aos resultados de assertividade, o SVM em conjunto com o histograma detectado com LBP acertou 74 de 296 amostras, tendo assim 25% de acerto, um F-1 igual a 0.25, e 109 de 652 amostras, tendo assim 16,71% de acerto, um F-1 igual à 0.167, nos *datasets* de teste da face toda e da parte ocular, respectivamente, o que leva-nos a concluir que a aplicação de filtros como equalização de histograma trazem uma influência negativa para a parte ocular e neutra para as imagens faciais.

Já o método VGGFace acertou 1 de 296 amostras, tendo assim 0,33% de acerto, um F-1 igual à 0.003, e 2 de 652 amostras, tendo assim 0.3% de acerto, um F-1 igual à 0.003, nos *datasets* de teste da face toda e da parte ocular, respectivamente, demonstrando, por sua vez, uma influência negativa para o modelo, em relação à aplicação de equalização de histograma.

As curvas ROC para os classificadores LBP são demonstradas a seguir, tendo que é possível observar uma evolução para ambas as curvas ROC com o uso de LBP.

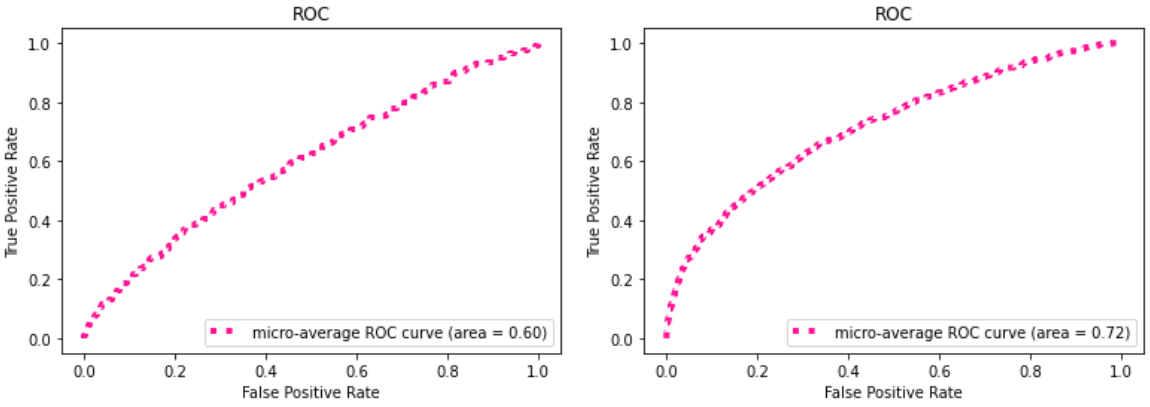

Figura 26: curvas ROC dos métodos LBP facial e ocular equalizado, respectivamente. (do autor).

Quanto às curvas ROC para os classificadores que utilizaram da VGGFace, pode-se observar na seguinte imagem (figura 27) que ambas obtiveram um regresso, ainda que não seja tão significativo.

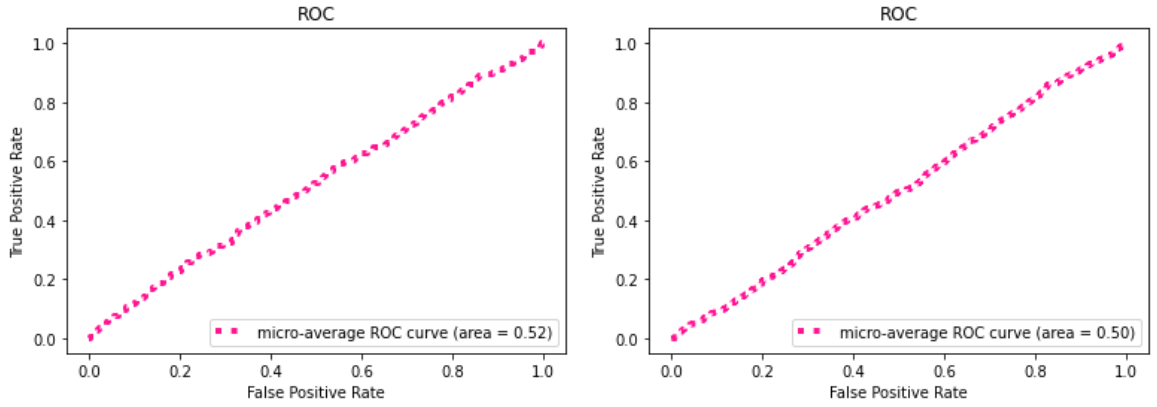

Figura 26: curvas ROC dos métodos VGGFace facial e ocular equalizado, respectivamente. (do autor).

Autenticação

Por fim, assim como descrito no rascunho da atividade, deveria ser desenvolvido um modelo que dado duas imagens, fosse possível indicar a autenticidade entre as mesmas. Para isso seria necessário utilizar de uma imagem base para ser considerada autêntica e que fizesse uma comparação com as outras imagens do *dataset* de modo que o modelo aprendesse as distâncias/diferenças entre as mesmas, diminuindo essa distância entre os valores autênticos e aumentasse entre os valores negativos. Algo que fica muito claro na visualização da seguinte imagem (figura 28) que representa a *Triple Loss*, muito utilizada para esse tipo de problema.

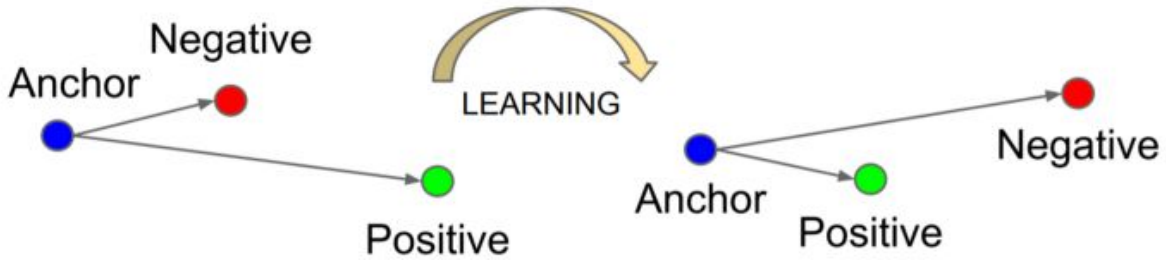

Figura 28: *Triple Loss* minimizando a distância entre uma âncora e uma positiva e maximizando a distância entre uma negativa. Disponível em: <https://medium.com/analytics-vidhya/triplet-loss-b9da35be21b8>. Acesso em dez. 2020.

Além das redes siamesas que também foram propostas para esse trabalho, para utilizar problemas *multiclass*, como é o caso deste problema, juntamente com os modelo de SVM que foram utilizados juntamente com o LBP, costuma-se aplicar o método *One versus all*, de modo que tenha-se um neurônio especialista para cada uma das classes o qual “diria” se aquele input é autêntico ou não.

Porém, dentre as possíveis alternativas para a resolução desse problema, a utilizada neste projeto contou com o auxílio da biblioteca scipy, em que com o método cosine é possível realizar a verificação calculando a distância cosseno entre a incorporação da identidade conhecida e as incorporações dos rostos dos candidatos.

O alcance dessa distância varia entre 0 e 1, tendo um valor limiar entre as identidades normalmente entre 0.4 e 0.6, sendo que nesse projeto o valor limiar padrão foi de 0.5.

Para encontrar qual o valor das diferenças de distância cosseno, as amostras passaram por um pré tratamento, seguido assim de uma predição por um dos modelos que foram desenvolvidos no decorrer deste trabalho e por fim, uma comparação.

Alguns dos testes podem ser visualizados na figura 29:

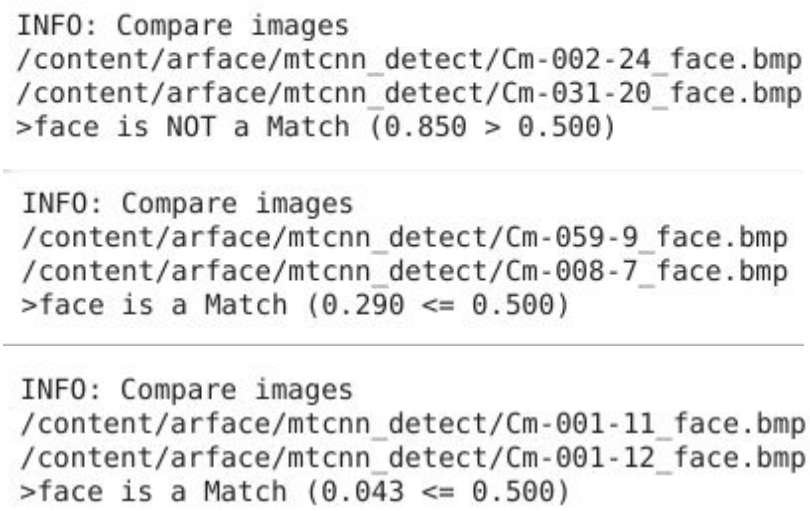

Figura 29: testes de autenticação entre duas imagens contendo um acerto, erro e acerto, em ordem. (do autor).

Referências

MARTINEZ, Aleix M. The AR face database. CVC Technical Report24, 1998.

ZHANG, Kaipeng et al. Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, v. 23, n. 10, p. 1499-1503, 2016.

OJALA, Timo; PIETIKAINEN, Matti; MAENPAA, Topi. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on pattern analysis and machine intelligence, v. 24, n. 7, p. 971-987, 2002.

PARKHI, Omkar M.; VEDALDI, Andrea; ZISSERMAN, Andrew. Deep face recognition. 2015.

CAO, Qiong et al. VGGFace2: A dataset for recognising faces across pose and age. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). IEEE, 2018. p. 67-74.