Universidade de Aveiro
2022

ANTÓNIO JOÃO
RODRIGUES
GONÇALVES

FOTOFACESUA: SISTEMA DE GESTÃO FOTOGRÁFICA DA UNIVERSIDADE DE AVEIRO

FOTOFACESUA: UNIVERSITY OF AVEIRO PHOTO MANAGEMENT SYSTEM

Universidade de Aveiro
2022

ANTÓNIO JOÃO RODRIGUES GONÇALVES

FOTOFACESUA: SISTEMA DE GESTÃO FOTOGRÁFICA DA UNIVERSIDADE DE AVEIRO

FOTOFACESUA: UNIVERSITY OF AVEIRO PHOTO MANAGEMENT SYSTEM

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Neto Vieira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho a todos pelo incansável apoio.

agradecimentos / acknowledgements

Agradeço desde já a todos os que me ajudaram e apoiaram na concretização desta tese. Um agradecimento especial aos orientadores da tese, Professor António José Ribeiro Neves e Professor José Neto Vieira, por me terem orientado, terem-se mostrado disponíveis para tirar as minhas dúvidas e dado sugestões e criticas construtivas ao meu trabalho. Quero também agradecer ao coolaborador Daniel Duarte Canedo por toda a ajuda e disponibilidade para tirar dúvidas e fornecer material para eu poder trabalhar. Um agradecimento à Professora Petia Georgieva Georgieva por toda a ajuda que me deu, a disponibilidade e as sugestões e críticas e também à minha colega Carolina Pinho por me ter ajudado a realizar os projetos de Aprendizagem Automática que foram uma mais valia para esta dissertação. Finalmente um agradecimento a todos os meus amigos, colegas e familiares que me apoiaram desde o inicio e que me incentivaram durante o ano letivo, principalmente os meus pais e aos meus avós.

Palavras Chave

Aprendizagem Máquina, Visão por computador, Aprendizagem Profunda, Aprendizagem por Transferência, Análise de Background, Deteção Semântica de Linhas

Resumo

Atualmente, a automação está presente em basicamente todos os sistemas computacionais. Com o aumento dos algoritmos de Aprendizagem Máquina ao longo dos anos, a necessidade de um ser humano intervir num sistema caiu bastante. Embora, em Universidades, Empresas e até Instituições governamentais, existam alguns sistemas que não foram automatizados. Um desses casos, é a gestão de fotos de perfil, que requer intervenção humana para verificar se a imagem segue o conjunto de critérios da Instituição que são obrigatórios para a submissão de uma nova foto.
O FotoFaces é um sistema de atualização de fotos do perfil dos colaboradores na Universidade de Aveiro que permite ao colaborador submeter uma nova foto e, automaticamente, através de um conjunto de algoritmos de processamnto de imagem, decidir se a foto cumpre um conjunto de critérios predefinidos. Uma das principais vantagens deste sistema é que pode ser utilizado em qualquer Instituição e pode ser adaptado às diferentes necessidades alterando apenas os algoritmos ou os critérios considerados. Esta Dissertação descreve algumas melhorias implementadas no sistema existente, bem como algumas funcionalidades novas ao nível dos algoritmos disponíveis.
As principais contribuições para o sistema são as seguintes: detecção de óculos de sol, detecção de chapéus e análise de background. Para as duas primeiras, foi necessário criar uma nova base de dados e rotulá-la para treinar, validar e testar uma rede de aprendizagem profunda por transferência, utilizada para detectar os óculos de sol e chapéus. Além disso, foram feitos vários testes variando os parâmetros da rede e usando algumas técnicas de aprendizagem máquina e pré-processamento sobre as imagens de entrada. Por fim, a análise do fundo consiste na implementação e teste de 2 algoritmos existentes na literatura, um de baixo nível e outro de aprendizagem profunda.
Globalmente, os resultados obtidos na melhoria dos algoritmos existentes, bem como o desempenho dos novos módulos de processamneto de imagem, permitiram criar um sistema de atualização de fotos do perfil mais robusto (melhoria dos algoritmos da versão de produção) e versátil (adição de novos algoritmos ao sistema).

Keywords

Machine Learning, Computer Vision, Deep Learning, Transfer Learning, Background Analysis, Sunglasses Detection, Hat Detection, Semantic Line Detector

Abstract

Nowadays, automation is present in basically every computational system. With the raise of Machine Learning algorithms through the years, the necessity of a human being to intervene in a system has dropped a lot. Although, in Universities, Companies and even governmental Institutions there are some systems that are have not been automatized. One of these cases, is the profile photo management, that stills requires human intervention to check if the image follows the Institution set of criteria that are obligatory to submit a new photo.
FotoFaces is a system for updating the profile photos of collaborators at the University of Aveiro that allows the collaborator to submit a new photo and, automatically, through a set of image processing algorithms, decide if the photo meets a set of predifined criteria. One of the main advantages of this system is that it can be used in any institution and can be adapted to different needs by just changing the algorithms or criteria considered. This Dissertation describes some improvements implemented in the existing system, as well as some new features in terms of the available algorithms.
The main contributions to the system are the following: sunglasses detection, hat detection and background analysis. For the first two, it was necessary to create a new database and label it to train, validate and test a deep transfer learning network, used to detect sunglasses and hats. In addition, several tests were performed varying the parameters of the network and using some machine learning and pre-processing techniques on the input images. Finally, the background analysis consists of the implementation and testing of 2 existing algorithms in the literature, one low level and the other deep learning.
Overall, the results obtained in the improvement of the existing algorithms, as well as the performance of the new image processing modules, allowed the creation of a more robust (improved production version algorithms) and versatile (addition of new algorithms to the system) profile photo update system.

Contents

List of Figures

List of Tables

Glossary

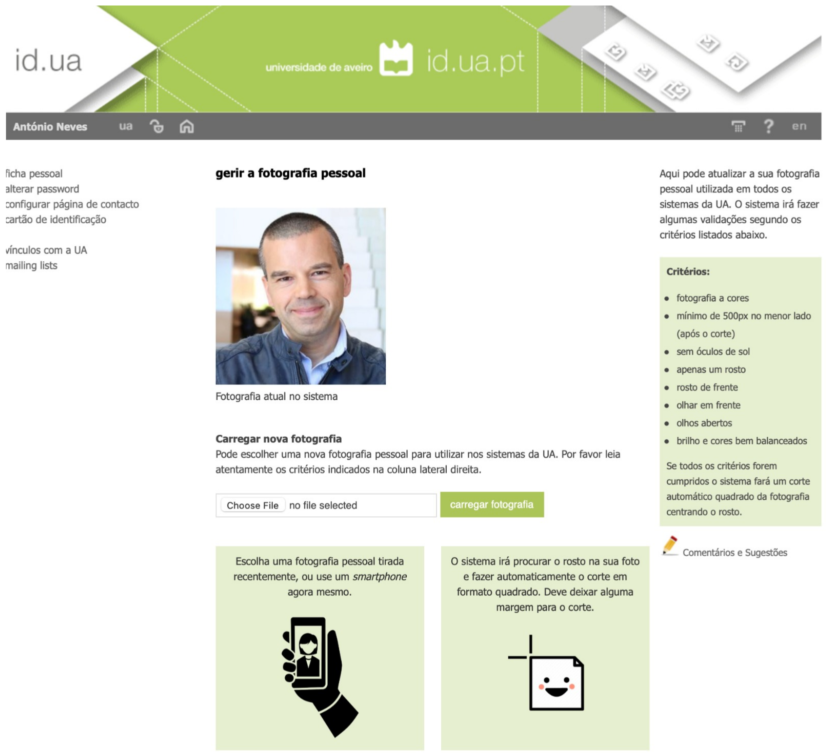
AFLW	Annotated Facial Landmark in the Wild
API	Application Programming Interface
BB	Bounding Box
BIQA	Blind Image quality Assessment
BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator
CK+	Extended Cohn-Kanade
CNN	Convolutional neural network
CPU	Central Process Unit
CV	Computer Vision
DL	Deep Learning
DLT	Direct Linear Transform
DNN	Deep Neural Network
DP2MFD	Deep Pyramid Deformable Part Model for Face Detection
DSC	Depthwise Separable Convolutional
EAR	Eye Aspect Ratio
FAR	False Acceptance Rate
FCNN	Fully Connected Neural Network
FDDB	Face Detection Data Set and Benchmark
FERET	Facial Recognition Technology
FPGA	Field-programmable gate array
FRCG	Face Recognition Grand Challenge
GAN	Generative Adversarial Neural Network
GCN	Graph Convolutional Network
GeLu	Gaussian Error Linear Unit
GGD	Generalized Gaussian Distribution
Google	Global Organization of Oriented Group Language of Earth
GPU	Graphics processing unit
Grad-CAM	Gradient-weighted Class Activation Mapping
HED	Holistical Edge Detection
HIoU	Harmony-based Intersection-over-union
HLW	Horizon Lines in the Wild
H-Net	Harmonization Network
HOG	Histogram of Oriented Gradient
HPE	Head-Pose Estimation
HSB	Hue, Saturation, and Brightness
HSV	Hue, Saturation and Value
h-swish	Hard Swish
ID	Identifier
IET	The Institution of Engineering and Technology
IJB-A	IARPA Janus Benchmark A

IoU	Intersection Over Union
IQA	Image Quality Assessment
IQS	Image Quality Score
KDEF	Karolinska Directed Emotional Faces
LIVE-itW	LIVE in the Wild
LUT	Look-Up-Table
MAE	Mean Absolute Error
mIoU	Mean Intersection Over Unit
ML	Machine Learning
MOS	Mean Opinion Score
MSCN	Mean Substracted Contrast Normalization
MSE	Mean squared error
MTCNN	Multi-Task Cascaded Convolutional Neural Network
MUG	Multimedia Understanding Group Facial Expression
NAS	Neural Architecture Search
NME	Normalized Mean Error
NN	Neural Network
NSS	Natural Scene Statistics
PAN	Path Aggregation Network
PCA	Principal component analysis
PLCC	Pearson Linear Correlation Coefficient
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RGB	Red, Green and Blue
RPN	Region Proposal Network
SGRH	Serviço de Gestão de Recursos Humanos
SEL	Semantic Line Dataset
SLNet	Semantic Line Network
S-Net	Selection Network
SROCC	Spearman Rank Order Correlation Coefficient
SVM	Support Vector Machine
TAR	True Acceptance Rate
VGG	Visual Geometry Group
WIDER	Web Image Dataset for Event Recognition
WWW	World Wide Web
Yahoo	Yet Another Hierarchical Officious Oracle
YCbCr	Luma, blue-difference and red-difference
YOLO	You Only Look Once

CHAPTER 1

Introduction

Fotofaces is an innovative system developed at University of Aveiro with the objective of allowing any user to change the profile photo without taking any human intervention by the Human Resources or Academic Services. This system uses several Computer Vision (CV) algorithms and Machine Learning (ML) models to evaluate a photo and produces several metrics regarding its properties. These metrics can be used by the user profile webpage to validate a new submitted photo. In Figure 1.1 can be seen the web page where it is possible to update the profile photo.



gerir a fotografia pessoal

Fotografia atual no sistema

Carregar nova fotografia
Pode escolher uma nova fotografia pessoal para utilizar nos sistemas da UA. Por favor leia atentamente os critérios indicados na coluna lateral direita.

Escolha uma fotografia pessoal tirada recentemente, ou use um *smartphone* agora mesmo.

O sistema irá procurar o rosto na sua foto e fazer automaticamente o corte em formato quadrado. Deve deixar alguma margem para o corte.

Aqui pode atualizar a sua fotografia pessoal utilizada em todos os sistemas da UA. O sistema irá fazer algumas validações segundo os critérios listados abaixo.

Critérios:

- fotografia a cores
- mínimo de 500px no menor lado (após o corte)
- sem óculos de sol
- apenas um rosto
- rosto de frente
- olhar em frente
- olhos abertos
- brilho e cores bem balanceados

Se todos os critérios forem cumpridos o sistema fará um corte automático quadrado da fotografia centrando o rosto.

Figure 1.1: Web page for photo submission

However, this system can be used in other institutions where the problem of keeping updated the personal directory exists, namely other educational institutions, companies, etc..

A prototype of this service is already in production at University of Aveiro. Figure 1.2 presents a simple diagram of the Fotofaces system in the context of the University of Aveiro website and Human Resources section. The system is composed by three parts: the website of the university, where it is possible to update the profile photo (`id.ua.pt`), the User that submits the new photo and the Serviço de Gestão de Recursos Humanos (SGRH) (the academic services of the university that only intervene in exceptional conditions).

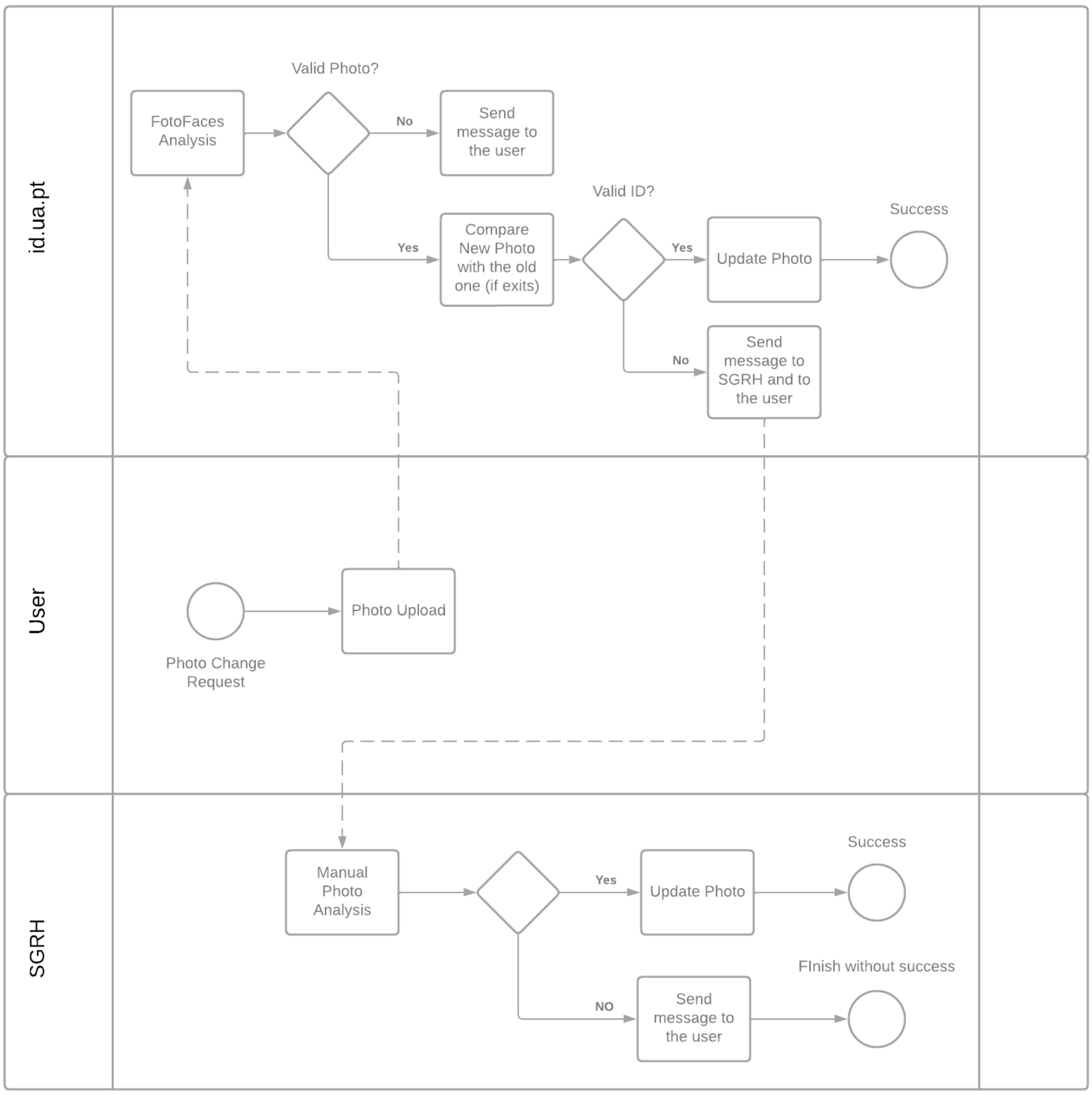


Figure 1.2: Procedure for submitting a photo

The main objective of this Dissertation is to improve the prototype of this system and develop a more robust one using new CV and ML approaches found in the current literature.

1.1 Software Implemented

As it was said before, there is a prototype test version of FotoFaces that follows some criteria to approve the photo of the candidate. The criteria that the photo needs to pass to be updated into the webservice are the following:

- It is a colour photo.
- The minimum size of the photo is 500 pixels in both dimensions.
- The photo has only one person in it.
- The face of the person should be entirely in the photo.
- The person in the photo can not have sunglasses.
- The brightness and colour should be balanced.
- The person in the submitted photo is the same as the one in previous photos in the system (if they exist).

The main objective of FotoFaces software is to guarantee that the photo fulfills the criteria described above.

The FotoFaces system tests these criteria using some CV methods. For example, to check if the photo has colors, it simply divides the photo into the 3 RGB channels and compares them. In the case that the channels are equal then the photo is not colored (grayscale photo), otherwise the photo has colors.

For the image size, it is simply used a code line to see if the width or the height of the image is higher than 500. In the case than one of then is lower than 500 pixels than the photo is not accepted.

Relatively to the person detection, the system detects the largest face in the photo, but do not tells the number of faces in the photo. This can be useful if there are persons in the background and it is done using a `Dlib` frontal face detector, that returns the Bounding Box (BB) of the largest detected face. Briefly, `Dlib` [1] is a C++ library with an extension for Python that uses ML algorithms and tools that are useful for real world applications.

In terms of the face recognition, if the face is too close to the camera or the person is not entirely in the photo, i.e, if the face landmarks are not inside the photo frame after cropping, then the system does not approve the photo. To do this, it is used `Dlib` 68 landmark detector and some threshold values that where obtained testing some photos in the system.

The sunglasses detection is done using a low-level method that basically, using the landmarks of the person in the photo, compares the nose region HSV color with the region below the eyes. If the difference between these regions is lower than a certain threshold than the person is not wearing sunglasses, otherwise the sunglasses are present. These two regions can be seen in Figure 1.3. Farther on, the eye and nose region will be defined in a different way to apply to other methods.

For image brightness the system basically calculates the average of the pixel intensity of the photo using RGB channels and then returns that value. If the average is lower than a certain threshold then it is considered that the image does not have sufficient bright and is not accepted. Also, it is used BRISQUE, that is a image quality classifier based on “online”

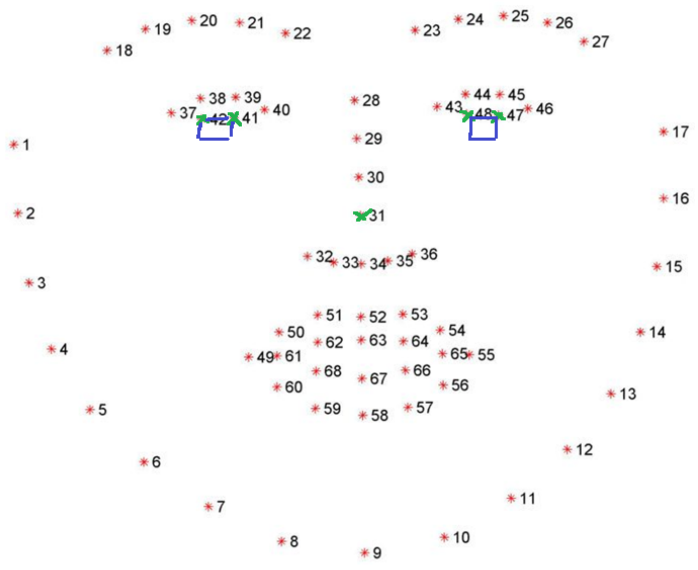


Figure 1.3: Eyes and Nose region used for the sunglasses HSV comparison method

responses. This method returns a value between 0 (best quality value) and 100 (worst quality value) that gives the quality value, and once again if the value is higher than a certain threshold the image is not accepted.

Finally to check if the candidate in the photo is the same as the one in the old photo it is used a `Dlib` face recognition Residual Neural Network (ResNet) model that is used to compute the face descriptor of each image. After is it that made the subtraction between the image descriptors, i.e., it is computed the distance between that images and if the distance is less then 0.6 than it is considered that the person in the new photo is the same as the person in the old photo.

For more details, check the Appendices where it is described with detail this prototype as well as the basic CV concepts in it.

1.2 Proposed Solution

In order to upgrade the FotoFaces system some features of the previous version were improved and some new features were added to the system that can be useful.

The list of changed/added features of the system is the following:

- Image Quality: added a recent model named Koncept224 [2]
- Sunglasses: improved the old version using DL models and constructing a new dataset
- Hat detection: added a DL model and created a hat detecting dataset
- Background Tilt Analysis: added some background tilt algorithms to compute the background tilt after applying the rotation to the candidate face

1.3 Structure of the Dissertation

The Dissertation is composed by 8 chapters in total where the first one is an Introductory chapter, the following 7 ones are the main contributions of the dissertation and the final one

(8) is the conclusions.

Excluding the first and the last chapters, all remaining ones have a similar structure, where the first section in each chapter is the literature review of that chapter topic and the next sections (in case that they exist) are the description of the work done.

CHAPTER 2

Image Quality Assessment (IQA)

IQA has been a fundamental aspect to determine the level of quality of different images taking into account all the difference factors that forms an image (capture, process, transmission, ...). In order to classify the images, a common term is used named IQA that can be of two types, namely subjective and objective.

Subjective methods, as the name suggests, are methods where the images are classified by humans using their stimulus and perception and then, using a large number of human classifications a score is computed typically named Mean Opinion Score (MOS).

Otherwise, objective methods are implemented using computational algorithms that can be of 3 types, namely Full-reference, Reduced-reference or even No-reference (Blind).

In the next section, are described some recent works that explore the subjective method Blind Image quality Assessment (BIQA) (No-reference method) to classify images.

More details about the image quality topic can be found in Appendices.

2.1 Literature Review

Vlad Hosu et al. [3] created the largest IQA database to date named KonIQ-10k with around 10k images. This database aims for ecological validity, authenticity of distortions, diversity of content and uses quality related indicators. It was made using crowdsourcing with 1,2 million quality ratings from 1.5k crowd workers. Also, the authors have proposed a DL model named KonCept512 to classify images using this new IQA database to train and test the model.

More specifically, the images from KonIQ-10k database were selected from 10 million YFCC100M ("largest publicly and freely useable multimedia collection") entries where the sampling algorithm uses 7 quality indicators and one content based on deep features. Also, each image has 120 ratings performed by the 1.5k crowdsourcing workers.

For the DL BIQA model fine-tuning was used over a InceptionResNetV2 architecture as the base of the model.

In terms of experimental results, the authors used the new proposed model over two datasets: KonIQ-10k and the LIVE in the Wild (LIVE-itW) dataset. For evaluating the model, were used 2 different metrics similar to the state-of-art works: Spearman Rank Order Correlation Coefficient (SROCC) and the Pearson Linear Correlation Coefficient (PLCC). Also, the authors have fine-tuned 5 models from the state-of-art works, initialized with ImageNet pre-trained weights, to get a comparison with their model.

First, the authors chose the best parameters for some transfer learning CNN models. Basically, they tested the models with different input image resolutions, in this case, 1024 $\times$ 768, 512 $\times$ 384 and 224 $\times$ 224, where they conclude that the best performance values were obtained for the 512 $\times$ 384 resolution images. After that, the authors used the best resolution image and trained the models again using 5 different loss functions, where the Mean squared error (MSE)-loss function outperformed the other ones. With the results obtained the best CNN model was the InceptionResNetV2 that was used to compare with the state-of-art methods.

Finally, the authors compared the KonCept512 model with the state-of-art methods, in this case 14 different models, and this new model got better results than the rest obtaining 0.921 SROCC and 0.937 PLCC values for the KonIQ-10k database and 0.825 SROCC and 0.848 PLCC values for the LIVE-itW database.

With all of this information, the proposed model and dataset seems to be good for IQA, more specifically, BIQA and this can be used as an improvement for the image quality submission on FotoFaces.

Weixia Zhang et al. [4] presents a new BIQA model that can determine the quality of images with different distortion scenarios: realistic and synthetic. Basically, this new model is constituted by 2 major steps that are IQA database combination and pairwise learning-to-rank model estimation. Also, for experimental results, the authors used 6 different datasets: LIVE, CSIQ TID2013, BID, LIVE Challenge, KonIQ-10K where the first 3 are synthetic and the other 3 are realistic.

The model used by the authors was ResNet composed by 2 streams. Each stream got a stage of convolution, batch normalization, Rectified Linear Unit (ReLU) nonlinearity and max-pooling followed by a set of layers from the bottlenet architecture that they used.

To train and test the model, the 6 datasets referred above were divided randomly into an 80%-20% split respectively. With this, the authors generated more 240 000 image pairs. For testing purposes, they have used the SROCC to quantify the performance over each dataset individually.

In terms of experimental results, the authors model was compared with 3 knowledge-driven BIQA models, i.e., do not need the MOSs for training, and 4 data-driven models. When comparing to these 7 models, the author´s model outperformed all, because they could not perform well over all datasets (synthetic and realistic) while the author´s model could because of the training over multiple datasets at the same time, that allows to adapt feature representations on both distortion scenarios. Also, the authors have used pre-trained weights on object recognition which prevents overfitting for individual databases.

Finally, they have trained the new model over a more challenging cross-database setting using image pairs from full TID2012 and LIVE Challenge. Again, this model outperformed the other 7 models. With this, it is possible to conclude that the model is able to perform well over both distortion scenarios realistic and synthetic. Also, this model only uses a single set of parameters, can be combined with other data-driven BIQA models (model agnostic) and it is easy to add new image pairs to train the model if new IQA databases emerges.

This work shown a method to compute the image quality over different distortion scenarios which could be used to improve FotoFaces system.

2.2 Koncept224 - KonIQ-10k Trained Model

As it was said in the Introductory chapter, the FotoFaces system has already implemented a IQA model named BRISQUE. To get a term of comparison with this model, it was added to the system one of the models from the literature review, in this case the Koncept224 model [2].

The choice to add the Koncept224 model was based on the fact that, the network used and its parameters, were available online and although the other was available too, we only managed to reproduce the Koncept224. Another important aspect is that the version of the Koncept model that is implemented, as the name said, is the one with 224x224 input image resolution that is not the best one that the authors proposed, but it was the one we managed to reproduce.

With all of that said, this Koncept224 model classifies the image between 0 and 100. if the image has got a 0 value, that means that the the image has the worst possible score. Otherwise, if the image has got a 100 score value, then it has the best possible score.

As it will be referred in some of the following chapters, the threshold defined for the IQA score with the Koncept224 model was of 15, while the threshold for the BRISQUE model was of 50. These threshold values where chosen by observing the score values of the different datasets used to test the FotoFaces system.

CHAPTER 3

Face Detection, Alignment and Recognition

This chapter presents a Literature review of three relevant topics used in the system: Face Detection, Alignment and Recognition. Although this can be a very interesting topic to explore and improve the system, it was chosen to prioritize other fundamental aspects of the system, like the Sunglasses and Hat Detection or the Background Analysis.

Face Detection, Alignment and Recognition has been explored since the emergence of of Image analysis and everyday, new works are published with new ideas and algorithms that innovates and overcome the existing solutions.

In this section, it is described shortly a survey about Face Detection give an idea of which are the current algorithms of this topic. Also, it was adopted a "historical" evolution approach in terms of papers chosen to present in the rest of this section for Face Detection, just to give an idea how these topic have evolved over time and to understand which were the problems and difficulties faced before, when DL did not existed. Finally, 2 reviews of Face Recognition and Face Alignment are presented that could be used in Future Work to improve the FotoFaces system.

A survey conducted by Chamandeep Vimal et al. [5] shows some of the "Face Detection's Various Techniques and Approaches".

The authors first do a Literature review of the most common Face detection techniques which are: Knowledge-based, Template matching, Feature invariant and Appearance-based and conclude that, generally the Appearance-based ones have the better results when comparing to the rest of the techniques.

After that, the authors describe 4 different approaches for Face Detection, namely: Face Detection Using Geometric Structure, Face Detection Approach Using High-Level Language, Face Detection Approach Using Haar-Like Features and Face Detection Using Facial Features.

Finally, it is made a comparison between the methods, in terms of advantages and disadvantages and also in terms of "OfKey Parameters", in this case, Learning Time, Precision, Ratio between detection rate and false alarm, and execution time.

In terms of advantages and disadvantages it was concluded that the Haar-Like Feature Base Face Detection has less false alarms and has an improved feature extraction part, but it requires much more time to execute and it is complex to implement. The Feature Based Face detection is more accurate than the other and has a low execution time, but at the same time has a high learning (training) time. The Geometric Based is an effective method and is easily implemented, but when comparing to the other technique has the lowest accuracy and consequently more false positives.

For the "OfKey parameters" (theoretical comparison), the authors only compared the 2 most suitable methods with these parameters (Haar-Like Feature base and Geometric based Face Detection). In this case, as it was said before the Haar-Like Feature based has a low execution time but in the same time has got more accuracy and precision, while the Geometric Based has a good/fast execution time but low accuracy and precision. Also, both of the methods have long learning time.

Rein-Lien Hsu et al. [6] propose an algorithm to detect faces in color images with uncontrolled lighting conditions and complex backgrounds. The algorithm proposed consists in two parts that are face localization to find face candidates and face feature detection to verify the detected faces.

The face location uses a "novel" lighting compensation technique to estimate and correct the color bias. Then the RGB components are transformed to a nonlinear Luma, blue-difference and red-difference (YCbCr) color space to detect the skin tone in the image using an elliptical skin model in the transformed color space. After that, the skin tone pixels are segmented using local color variance into connected components that are grouped into face candidates depending on the spatial arrangement of these components and the similarity of their color. With the skin group regions detected then the facial features, eyes and mouth, are extracted using the feature maps (eye maps and mouth maps) derived from the luma and the chroma of the image.

To test the algorithm the authors created a new dataset for face detection using MPEG7 videos, the World Wide Web (WWW) and personal photo collections with a total of 206 images. All these images have different backgrounds, indoors and outdoors and were taken under different uncontrolled environment lighting conditions. Also, the faces present in the images contains different positions, scale, orientation, 3D poses and facial expressions.

It was concluded that the algorithm could detect multiple faces with different sizes with a large range of facial variations in an image. Also, because of the nonlinear transformation of the YCbCr color space the algorithm detects different skin-tones (darker and brighter). Other face aspects like different facial expressions, people wearing glasses, non-frontal face positions and facial hair are not an obstacle and the system can still detect the faces.

With all that, the authors obtained for the HHI MP4G7 dataset after the 2 stages (without facial feature extraction) 97 % of accuracy for face detection, and after third stage (with facial features extraction) the detection rate decreased to 89.4 % for frontal-faces and 74.67 % for half-faces profiles. The decrease is easily justified for the simple fact that the eyes and the mouth are not visible. To note that after feature extraction the number of false positives is reduced a lot.

Also, they have tested the algorithm over the Champion dataset (227 images) that is composed mostly by frontal and near-frontal faces. They achieved 99.12 % detection rate for the skin groups and 91.63% for the facial feature detection.

Finally, they tested the algorithm over some family photos as well as some Yet Another Hierarchical Officious Oracle (Yahoo) news site. The detection rate over that new dataset (382 images) was of 80.35 % with a 10.41% false positive rate.

Although this algorithm detects faces over different poses, with uncontrolled environments and with different skin tones, this work is from 2002 and is non-neural based, which most of the recent works are based on and got better accuracy values, so it will be not considered to get implemented in FotoFaces system.

Huaizu Jiang et al. [7] proposed the use of the Faster Region-based Convolutional Neural Network (R-CNN) framework into a large new face detection dataset named Web Image Dataset for Event Recognition (WIDER). Faster R-CNN has got advantages when comparing to non-neural based methods because it can automatically extract features and has some advantages when comparing to neural based methods like allowing the end-to-end learning of all layers, making the network more robust.

To train the Faster R-CNN, the authors used 12880 images with 158 424 faces of WIDER dataset. These images contain faces that vary in scale and pose and the number of faces per image is not the same. The face detection Faster R-CNN is based on the ImageNet model, VGG16. Also, the authors use other face detection datasets like Face Detection Data Set and Benchmark (FDDB) and IARPA Janus Benchmark A (IJB-A) to test the network.

Moreover, they compared different face detection approaches with the Region Proposal Network (RPN), in this case the base for Faster R-CNN architecture, like EdgeBox, Faceness and DeepBox over the FDDB dataset and it was concluded that RPN is better than the rest because they use a deeper CNN when comparing to DeepBox and do not rely on other object proposal methods like EdgeBox.

After that, they compared the Region based CNN methods, i.e, R-CNN, Fast R-CNN and Faster R-CNN using the FDDB dataset. In this case the Faster R-CNN outperformed the other 2 methods. One of the reasons is that the RPN base implemented in Faster R-CNN is a deep CNN. Also, Faster R-CNN proved to be much faster than R-CNN and Fast R-CNN with a running time of only 0.38 seconds. The other two architectures, R-CNN and Fast R-CNN had best running times of 14.81 seconds and 2.94 seconds respectively.

The authors compared the Faster R-CNN with the state-of-art top detectors of the FDDB dataset. In terms of discrete scores, the Faster R-CNN achieved the best results when there were more than 200 false positives in the entire test set. Although for continuous scores, the Faster R-R-CNN could not be the MultiresHPM method. To note that, the discrete score is when the detection is considered to be positive if its Intersection Over Union (IoU) ratio with its one-one matched ground-truth annotation is greater than 0.5, and the continuous score is when the true positives are weighted by the IoU scores. This is easily explained by the fact that the results of the Faster R-CNN are not always around the annotated images. With that, for 500 false positives the Faster R-CNN obtained 0.952 and 0.718 values for true positive rate continuous and discrete values. The WIDER dataset was divided into 3 detection difficulties using the EdgeBox detection scores. The harder the difficulty the worst was the performance of Faster R-CNN comparing to the other detection models, but for the easiest difficulty the Faster R-CNN got the best values. Also, it was seen that Faster R-CNN was capable of dealing with challenging cases where there were overlapping faces and faces with extreme poses and scales. The main disadvantages of Faster R-CNN were when using images that were really crowded that result in some false negatives.

Finally, the authors used a supervised and an unsupervised annotation style adaptations over the IJB-A dataset and the results obtained were the best when comparing to other methods. With this work it was concluded that the effectiveness of the Faster R-CNN was from the RPN base architecture. Also, due to the sharing of the convolutional layers between the 2 architectures the computational load was decreased. The Faster R-CNN, which is an architecture used for general object detection, proved to be very good to detect faces.

This work seems good to detect faces using a CNN architecture for face detection and reduces the computational load comparing to other used architectures.

Meihua Gu et al. [8] proposed a novel classroom face detection algorithm based on an improved version of Multi-Task Cascaded Convolutional Neural Network (MTCNN). The authors use a deep residual feature generation network to "upgrade" the MTCNN model (named as MTCNN_v2) that will allow to detect small-objects, in this case faces, generating "super-resolved large-object". After that, they apply an up-and-down crop technique to solve the problem of crowded classroom and uneven scaled faces.

To train the MTCNN_v2 the authors used the ML TensorFlow framework and the WIDER FACE dataset [8]. To test it the FDDB dataset was used. First they train and test the new model without the up-and-down strategy using evaluation indexes defined as ContRoc and DiscRoc. The scores were compared with other advanced face detection methods like Cascade CNN, Deep Pyramid Deformable Part Model for Face Detection (DP2MFD) and Faceness. The new model MTCNN_v2 obtained the better results with a detection accuracy of 96.01% which is better 1 % than the MTCNN model. Also, the MTCNN_v2 had a good accuracy over occlusion and faces with different poses while the other methods failed in most of these cases. The MTCNN_v2 was the model with the highest speed which proves it utility in real classroom scenario. Finally, it was evaluated the MTCNN_v2 with the up-and-down cropping method and the performance was even better, with more 1.4% than the previous MTCNN_v2 model. To note that the results were tested over a real video of classroom scenario, which proves the efficiency and accuracy of this new model.

This face detection work is suitable for real-time face detection applications and has good detection rate over occlusions and different head poses estimations.

Yanda Meng et al. [9] proposes a novel "3D Dense Face Alignment with Fused Features by Aggregating CNNs and Graph Convolutional Network (GCN)s". The authors use a multi-level aggregation network to obtain the coordinates of 3D vertices face from a 2D image using an end-to-end method, combining CNNs and GCNs fusing their features. With this, it is demonstrated that low-level semantic information and high-level spatial features are enough to estimate 3D facial geometry. To note that GCNs works well on non-grid data like 3D meshes and with that it is possible to avoid or reduce the 2D image noise. With the new model the authors can have a 3D face reconstruction and a dense face alignment with a relatively lower size model. The network proposed by the authors is constituted by an encoder and a decoder that are connected by several nested residual convolutional blocks (aggregation block). Also, the authors proposed a new loss function that prevents the model from taking large update steps when reaching small errors for late training stages and who can adapts quickly in early training stages when having a large error.

In terms of experimental results, the model was trained over 300W-LP dataset. For sparse face alignment and dense evaluation purposes, the Annotated Facial Landmark in the Wild (AFLW)2000-3D dataset was used and for only 3D sparse face alignment an extension of AFLW2000-3D dataset was used, named AFLW-LFPA. Finally, the Florence dataset was used to evaluate the face reconstruction part. To note that a comparison was done for all the evaluation methods with state-of-art works.

With all of that, for face alignment (dense and sparce evaluations) the results were compared using Normalized Mean Error (NME) that is an alignment accuracy metric. "NME is the average of the landmarks error normalized by the size of the bounding box", defined as the rectangle hull of the 68 landmarks. The model proposed by the authors obtained better performance than the rest of the state-of-art works with 19% higher performance over the best model from these state-of-art works. To note that, one of the advantages on face alignment task when comparing to the other models is the fact that this new model gains more information over the visible face parts which allows the GCNs to get better regression values for the unseen (invisible) mesh vertices.

For 3D Face Reconstruction the metric chosen was the MSE nomalized by outer interocular distance of 3D coordinates. Again, this new model outperformed the state-of-art methods by a large margin. In this case, this new model could cover lateral regions of the face, like the ears and necks, while the state-of-art methods could not.

Moreover, the authors did an ablation study over the Aggregation Block, the Parameters of the Loss Function, and the Loss Function.

Finally, the authors tested the running speed of the network achieving a 16.0 millisecond time of running speed over their machine (specified in the paper) which is a comparable value with the remaining state-of-art methods. Also, they compared the size of their network with the state-of-art methods and this model is the smallest one with "only" 84.5 MB.

This paper presented an innovative face alignment and 3D face reconstruction lightweight algorithm that could be used to improve the performance of FotoFaces system.

Sharma S, Karthikeyan Shanmugasundaram et al. [10] developed a new "FAREC - CNN Efficient Face Recognition Technique using `Dlib`". The system developed consistes of 4 processes: face detection, face alignment, face cropping and feature extraction.

In terms of face detection, they had used "computer vision technology" to detect the frontal faces, which is not helpful at all because they do not specify how this face detection is made. For the face alignment the authors used the `Dlib` package with an effectiveness of the alignment up to 45 degrees. The face cropping was made around the outer pose estimation of the face using `Dlib` shape predictor. Finally the feature extraction was made using a simple CNN architecture that can be seen in the paper [10].

The Face Recognition Grand Challenge (FRCG) dataset was used to train, validate, and test the new FAREC system. This dataset got around 50k digital images (environment controlled and uncontrolled) which is a relatively high number for face recognition datasets.

For evaluation purposes the authors used True Acceptance Rate (TAR) and False Acceptance Rate (FAR). These rates are normally used for face recognition models, where the TAR is used to measure the face recognition that is correctly made and the FAR is used to measure the likelihood the face recognition system will have to incorrectly detect a face.

The results obtained by the authors were 96% of accuracy for FRCG with a FAR of only 0.1% which are better results than the state-of-art methods named by the authors.

One of the disadvantages of this work is that is not specified the face detection methods. Although, this could be combined with some of the previous described works above and develop a novel Face Recognition algorithm.

CHAPTER 4

Head-Pose Estimation (HPE)

In this chapter, as in the previous one, a Literature review section is shown where it is described a short review from a recent survey and also described 2 recent interesting works about HPE that are not referred in that survey.

A survey conducted by Andrea F. Abate et al. [11] shows several applications and techiques of HPE of the last 5 years as well as the different datasets available for HPE.

The authors identify 3 main input data types: Deph Images, 2D images and videos. These types changes the types of applications where HPE can be applied.

With that information, the different datasets that exist are also divided between these 3 types of data. BIWI Kinect Head Pose Database, ICT-3DHP, ETH Face Pose Range Image Dataset are some of the datasets for deph images. CMU-MultiPIE, The AFLW, AFLW2000 and 300W_lp are datasets for 2D images. UPNA Head Pose Database, Boston University Head Pose Database, EYEDIAP Database are some of the datasets for video input. To note that the authors described with more detail each of these datasets and also show more datasets examples for each input data too.

Next, the authors show the different existing techniques for HPE. These techniques are divided into 2 main groups present in HPE that are the training and training-free methods. As the name suggests, the training methods require to re-train, re-build the model in case that there are some type of changes in the system requirements or in the datasets for example. On the other hand, the training-free methods do not need to be re-trained and in case they need to be re-modeled normally it is a fast task.

Finally, the authors made a comparison between these 2 types of methods (training and training-free) and, despite there are more training methods, it was not possible to conclude that the training methods are better than the training-free ones.

With all of that, this survey gives us an idea of the recent techniques and datasets available for HPE as well as the type of applications where these techniques and datasets can be applied.

Xiao Li et al. [12] present a new approach to get an accurate HPE using image rectification and a lightweight CNN.

In simple terms, the image rectification is the correction of the perspective distortion of the face image that occurs because of the misalignment of the face and camera coordinate system. To note that this distortion is higher if the face is farther away from the camera. To do this image rectification, the authors use rotation to transform the real camera image plane into a new virtual image plane aligned with the face center. After that, the rectified image is given as input to the lightweight network to estimate the Head Pose values, in this case, Mean Absolute Error (MAE), pitch, roll and yaw values. Finally, the authors transform again the virtual image plane (coordinate system) into the real camera image plane (coordinate system), to obtain the values for the head pose into the real camera coordinate system.

The lightweight network used was a simple CNN that uses depthwise convolution and pointwise convolution techniques. Basically, these techniques allow the CNN to achieve high performance values with a good tradeoff between the accuracy and computational costs. In terms of activation functions, the authors used ReLU and Hard Swish (h-swish) to improve the performance of the network. Also, to compute the Euler Angles 66 intervals and a softmax layer were used to predict the probability that a certain image belongs to a certain interval. The sum of the midpoint value of each interval is then the final Euler Angle value estimation.

With all of that, the authors did several experiments with the new approach and compared with the state-of-art methods using 3 different datsets: BIWI, 300W-LP and AFLW2000. To note that BIWI is the only dataset that contains the ground truth head pose calculated from 3D information while the other 2 datasets use the 2D information to calculate the head-pose.

The first experiment was to compare the accuracy and model size of the new approach using the BIWI dataset for training and testing the network. The results obtained for MAE, roll, pitch and yaw were better than the rest of the state-of-art methods, with one exception on the Pitch value, where a method got better accuracy. This is easily explained by the fact that the state-of-art methods used RGB images and depth information while this approach by the authors only used the RGB images. Also, the new approach was the one with the lowest model size.

The second experiment was similar to the first one but now using the BIWI dataset only to train the network and the 300W-LP to test it. It obtained the best results for the Yaw, Roll and MAE with the exception for the Pitch error where the Img2pose state-of-art method got a small error. Again, the model size of the new approach was the smallest. Also, the authors trained and tested one of the state-of-art methods with image rectification and compared with the results without it. An improvement was observed of around 15% of MAE compared to the previous one, and the pitch, roll and yaw errors also have decreased. To note that in the second experiment a state-of-art method was used that applied the `Dlib` library to compute the HPE, i.e., the same method used in FotoFaces system and it was one of the worst methods in the comparison table in this paper.

The focus of the third experiment was to see the improvement of the network with the proposed discriminative loss function using the 300W-LP dataset for training and the AFLW2000 dataset for testing. The results obtained showed that the model size and the MAE was the lowest, but the pitch, roll and yaw errors were surpassed by the Img2pose method. Although, the Img2pose method got a model size around 188 times higher than the new approach. With this experiment the authors proved that the model works well for large head poses with different illumination conditions.

The fourth experiment was to observe the processing speed of the lightweight network where each state-of-art method was tested with the same conditions and evaluated 100 times. The authors approach obtained the fastest processing speed of them all with 2910 Images/sec using the Graphics processing unit (GPU) and 161 Images/sec using the Central Process Unit (CPU). With the model´s size of this approach and the processing speed, this can be implemented over embedded applications that uses small GPUs, like Field-programmable gate array (FPGA)´s for example.

Finally, the authors made another experiment to see the effects of image resolution and data augmentation over the 300W-LP dataset. The results showed that with higher resolution the results for MAE, roll, pitch and yaw were better and applying more data augmentation techniques, i.e, cropping, random scale, downsampling also improves the performance of the network.

This work presented an innovative and lightweight approach to HPE which can be considered as a new approach to implemented over the FotoFaces system, since it is quick and obtained good results comparing to the state-of-art methods where is also referred the FotoFaces HPE prototype method.

Naina Dhingra [13] presents a lightweight network for HPE that is suitable for mobile devices. The author proposes a LwPosr network that uses Depthwise Separable Convolutional (DSC) with transformer encoder layers that have two streams with 3 stages each that allows fine-grained regression for predicting head poses. The final HPE values are given by the mean of the 3 HPE in each stage.

DSC is a technique that factorizes the conventional convolutional layer into a depthwise and a pointwise convolutional layer which means that the computational power is reduced, and the model size is lower.

The transformers encoders are used to get global information with their self-attention mechanism. In this paper the author uses a transformer encoder with position embeddings. In simple terms, without position embedding the encoder is permutation-invariant and does not use the spatial information of the features.

For experimental results, the author uses 3 HPE datasets: 300W-LP, AFLW2000 and BIWI dataset.

Two “experiment protocols” were made to compare with the state-of-art methods. The first one uses the 300W-LP dataset to train the network and the testing phase is performed over the AFLW2000 and BIWI datasets. The second protocol uses only the BIWI dataset to train and test. Both of them uses MTCNN as face detector.

In terms of experimental results, for the first protocol the author obtained the model with the lowest size and less parameters. Nevertheless, some of the state-of-art methods got better results for the MAE in both datasets but with a tradeoff in term of parameters, in this case, around 163.5 times more parameters for AFLW2000 dataset and 27.7 times more parameters for BIWI dataset. When comparing with the other lightweight networks the author´s network got the best results. Also, a LwPosr α network was trained and tested where the only difference with the initial one was the activation function of the transformer encoder, in this case, it was used the Gaussian Error Linear Unit (GeLu) activation function instead of the ReLU. For the second protocol results the author´s network got the lowest number of parameters too. Again, there were networks where the HPE values were better but with a higher number of parameters which can difficult the implementation over mobile devices. Comparing with the lightweight networks, the values have improved.

Finally, the author has made some experiments with different parameters of the network, such has the number of layers, heads, activation functions, position embeddings in the transformer encoder, learning rates, weights in the weighted mean, loss functions and number of stages in the LwPosr. It was concluded, when increasing number of stages, that the results were similar and the number of parameters were higher, so does not make any sense to add more stages over the network.

With all of that, this paper provides an innovative lightweight network with transformers encoders that is suitable to compute HPE on mobile devices due to the reduced computational power load and the model size of the network.

CHAPTER 5

Sunglasses detection

Nowadays, ML models are present in many computational systems and they are very efficient and accurate. Taking into account our research, when looking at Sunglasses detection recent works, all of them have been made using some basic CV methods, for example, detect sunglasses comparing the nose region color with the region below the eye where normally sunglasses are present. One of the main reasons that this happens, not only in sunglasses detection but also in other CV topics, is the lack of datasets to train, validate and test the ML models.

The present chapter has with some literature reviews of similar works, like eyeglasses (glasses) detection, and some ML techniques that can be useful to improve the model accuracy and understand the functioning of the network, like data augmentation and attention maps.

In the remaining subsections, ML approaches are shown, some combined with low level techniques, as well as some examples from the new sunglasses dataset.

5.1 Literature Review

A survey conducted by Shorten et al. [14] refers that data augmentation is a powerful tool to build useful DL models where the validation error must continue to decrease with the training error. This article focuses on how data augmentation can reduce the overfitting over the training set over DL models.

This article explains and compares numerous ways and considerations on data augmentation. The most common techniques are: random erasing, cropping, zoom in/out, rotation, adding noise, Geometric transformations. There are also more complex augmentations by applying metalearning concepts from Neural Architecture Search (NAS) [15] to Data that results in works such as Neural Augmentation, Smart Augmentation, and AutoAugment.

Relatively to traditional data augmentation it is concluded that it is not a linear science comparing techniques, as it depends highly on the specific problem and data set and it is important to maintain a balance between complexity and results of each one. Some can point out that cropping, flipping and rotation perform better than others.

It is also concluded that combining augmentations techniques such as cropping, flipping, color shifts, and random erasing can result in massively increased dataset sizes. However, this is not guaranteed to be advantageous. In domains with very limited data, this could result in further overfitting.

Daniel Canedo et al. [16] show the impact of preprocessing algorithms over DL models, in this case facial expression recognition. In this work, the authors focus is to apply several preprocessing methods over a simple CNN model composed only by 2 convolutional layers and see the impact of this preprocessing methods using Gradient-weighted Class Activation Mapping (Grad-CAM). Grad-CAM is a technique that allows to observe what the convolutional layers of the network are choosing as “features” to classify the input of the network.

To train the CNN the authors used the Extended Cohn-Kanade (CK+) database which is a database with controlled environment and posed facial expressions. For cross-database testing they used Karolinska Directed Emotional Faces (KDEF) database and Multimedia Understanding Group Facial Expression (MUG) database, because using the same database can cause overfitting of the model and therefore the convolutional layers do not learn the appropriate features. Also, for testing purposes, 6 facial expressions labels (anger, disgust, fear, happiness, sadness, surprise) were used plus the neutral expression. These facial expressions were used in others similar works. To note that the objective in this paper is to see the impact of the data preprocessing and not achieving high accuracy results for facial expression recognition.

In terms of data preprocessing, the authors applied rotation, cropping, intensity normalization, histogram equalization and smoothing (guassian filter) before training the model. The CNN was trained for 50 epochs each time a new pre-preprocessing method was added to the model and 90% of each database was used for training and the remaining 10% to test the CNN. Also, a 10-fold cross-validation was done for each training set.

After testing, the authors made an averaging of the attention maps for each facial expression for the raw set and the smooth set (includes all the preprocessing methods) in order to demonstrate the power that preprocessing can have in facial expression recognition, and, more generally, in ML models. Also, with preprocessing the accuracy values have improved around 18% which is a lot for such a simple CNN. The preprocessing method with more impact in this task was the cropping, because it reduces the background noise and gives emphasis to the face of the person for this task.

Finally, the accuracy value for the smooth set was 93.90% which is competitive with related works about facial recognition.

Jorge Bueno Perez in [17] presents a Transfer Learning implementation using MobileNetV2 trained model. MobileNetV2 is a model developed and trained by Global Organization of Oriented Group Language of Earth (Google) with 1.4 million of images and 1000 classes and it receives images as input with pixel values between -1 and 1. The author uses an artificial dataset of people with and without glasses generated by a Generative Adversarial Neural Network (GAN). In his work, the author made some data cleaning to the dataset, rescaled the pixel values of each image to values between -1 and 1 and divided the dataset into training

, cross-validation and test subsets. Also, he performs data augmentation, rotating each image two times in order to get better results and reduce overfitting of the training subset. Finally, he freezes the layers of the MobileNetV2 to make feature extraction and adds a dense layers to the end of the model that converts this features into a single prediction.

With all of that, for the first 10 epochs the accuracy values for the training and validation subsets are around 90% which are not good comparing to others works, so the author does Fine-Tunning, unfreezing the top layers of the model and training again the network for more 5 epochs using the previous trained parameters as initial start point. The results that he obtains are astonishing with a test accuracy of 100%, training accuracy of 99.7 % and a cross-validation accuracy of 99.6%. Although these results are really impressive, it is worth noting that this database is a controlled database, so this model only works well for controlled images.

Saddan Bekhet et al. [18] present a more complex work of Glasses Detection made at The Institution of Engineering and Technology (IET) where "a robust DL approach for glasses detection in non-standard facial images" is developed.

The objective of this work is to detect glasses or no glasses in non-standard photos, in this case, in selfies in real-life environment, i.e., with shadows and bad lighting conditions as well as non-centered selfies and other difficult factors when talking about detection in CV. The authors propose a CNN model, using transfer learning in combination with new learned feature maps in order to compensate the uncontrolled nature of the dataset of selfies used. The dataset is composed by 46836 annotated images with different labels like glasses, sunglasses, lipstick and others but, as the authors said, the relevant ones are glasses/no glasses.

The CNN architecture is the AlexNet structure. AlexNet is a CNN that was trained with more than 1 million images. This network is composed by 8 layers: 5 convolutional layers and 3 fully connected layers and can classify up to 1000 different objects, for example a keyboard, mouse, pencil, etc. The input for AlexNet are 224x224 resolution images but the authors changed this to 227x227 in order to fit with the dataset. Also, they "replaced" the last 3 layers (the fully connected layers) to predict the glasses/no glasses labels. This was made adding a fully connected layer, a softmax layer and a classification output layer with 2 labels (glasses/no glasses). This fine-tuning approach allows the model to get specifics and bias of the selfies photos based on the features obtained by the first layers of the CNN AlexNet. Moreover, the authors added 2 dropout layers with 50% random dropout to prevent overfitting, because the dataset used was not too big. Also, the inputs that are not set to zero are scaled up by 1 unit making the sum of all inputs unchanged.

In terms of data augmentation, the authors, made an online (runtime) approach doing random translations in the (-30,30) interval to avoid positional bias in the data because most of the selfies were centered, using a label-preserving transformation.

The authors trained this model using stochastic gradient descent with a batch size of 10 examples, momentum of 0.9 and weight decay of 0.0004 (this is a small value to reduce the training error and facilitate the model learning). The freezed layers did not change the weights. The whole process took around 15 days using an augmented selfie dataset with around 100k images. To note that the authors used 70% of the images for training and 30% of the images for validation. With that, they measured the accuracy of the network and calculated the log-loss obtaining the value of 96% of accuracy and 0.15% of log-loss.

The authors then compared their results with recent glasses detection proposed models with uncontrolled environment images and they got the highest accuracy of all of the works.

Also, they have studied their model (training and validation) with many other datasets with uncontrolled environment images obtaining accuracy values between 84.6% and 97.43% for the datasets which proves the robustness of the proposed network.

Finally, they have done the training and validation of their "new" architecture without freezing any convolutional layer, i.e., computing the parameters of each layer. The results were of 85,6% of accuracy, which is really good taking into account that their dataset is about 4.6% of the dataset used for the AlexNet and has just 9.5% less of accuracy when comparing to the previous trained model with transfer learning.

This paper shown the glasses detection in uncontrolled environment and, although it does not have a 99% or above accuracy like in the Jorge Bueno Perez et al. [17], is it for sure more robust because it takes into account a lot of factors in images that the other do not cover.

Basbrain et al. [19] presents a method for eyeglasses detection in facial images. It is based on extracting deep features from a "well-designed" shallow CNN with less complex structures that can be used in different facial analysis system frameworks without consuming their resources and, furthermore, achieves high accuracy in real-time. The proposed Shallow-GlassesNet was inspired by GoogLeNet architectures and contains six layers: three convolution and two max-pooling layers followed by a fully-connected layer.

The kernel size and stride of Conv1,Conv2 and Conv3 layers are set as 7×7 (2), 1×1 (1), and 3×3 (1), and their outputs are feature maps of sizes 64, 64 and 192 respectively. Each of these layers has similar corresponding layers in GoogLeNet.

When designing the CNN, a GoogLeNet was fine-tuned with a small facial database in order to avoid some gradient instability caused by bad initialization of the network weights. To select the best performance layer according to its accuracy and the time consuming, the authors used each layer of the fine-tuned GoogLeNet to extract features of the validation set images to test its performance. Then, when using the Shallow-GlassesNet, its three convolutional layers were initialised with the weights of the corresponding layers of the fine-tuned GoogLeNet.

Then, by utilising the Shallow-GlassesNet as feature extractor, the resulting image features are normalised and fed into a Linear SVM classifier, in order to increase the accuracy, which is trained on the used database (referred below).

To sum up, the proposed pipeline consists on three parts: face detection, features extraction by Shallow GlassesNet, and classification/detection by the SVM.

The authors adopted the USTC-NVIE [20] database, however, this database is small for training the SVM and in order to deal with overfitting, data was augmented using different techniques of data augmentation such as horizontal and vertical rotations.

The framework proposed in this paper demonstrate an excellent performance which achieves a mean accuracy of 99.78% with fast results (0.0782 s/Image).

Wu, et al. [21] propose a novel method for glasses detection in real-life photos where glasses detectors are learned by using a variation of a boosting algorithm, called real Adaboost. The main objective is to boost Look-Up-Table (LUT) type weak classifiers based on simple wavelet features that the authors designed [22], instead of the typical naïve threshold-type weak classifiers. Then, from these weak classifiers, a strong classifier is boosted. Wavelet scattering was a famous technique for signals that was adapted for image classification and inspired Wavelet scattering Network that can be referred as a deep network because it performs convolution, non-linearity and pooling[23]. Some researchers say that at some point this technique has a better performance on feature extraction than Principal component analysis (PCA) methods for example [24]. The authors also explored two types of wavelet features, Haar and Gabor. With this Boosting algorithms the performance of this solution working interactively resulted in an accuracy of 98.4% in 0.09ms (time per sample) tested on collected images from Facial Recognition Technology (FERET) [25] data set.

S. Asteriadis et al. [26] present a novel algorithm on Eye Detection and Location using geometrical information of the image that can be applied to low resolution images, where pixel intensity information might be not reliable and the details of the eyes sometimes are not present.

First, the Eye Region detection is made. In short, the authors use standard PCA method to get the eigenfaces values to detect the face in the image. Then they apply Canny edge detector to get the edges in each pixel of the image. With these edges it is computed a respective vector that points to the closest pixel and then, instead of using the intensity values, they compute the magnitude (length) and the slope (angle) of each vector in the image. PCA is applied once again, but now to derive the eigenvectors of each eye image that is used for training. Finally, it is projected the magnitude and slope values of the candidate image over the training spaces spanned by the eigenvectors. The similarity of the projection weights of the candidate image with the training images is used to detect the eye presence.

For the Eye center location, the facial areas (eye area) are used that have the smallest distances to the training model eyes computed in the previous step. To note that, the authors used the approximate position of eyes on the face above, in order to make the algorithm faster. Then Canny edge detector is applied with different parameters than the previous ones to compute even weak edges. With that it is possible to handle cases where the person is wearing glasses because glasses causes several reflections and can make the eye characteristics less visible. Now that the edges are computed, they use the information in the 15x25 pixels of the lower part of the eye area that gives most of the information about the eye and does not take into account the eyebrow or the glasses skeleton and, with that, it is searched for 3 horizontal and vertical lines that have most intersections with edges. The intersection of the horizontal line that have the medium number of edge intersections among the 3 lines chosen and the vertical one with the same characteristics, gives the best approximation for the eye center. Some refinements can be done in this algorithm like applying the horizontal and vertical line method again around the previous point of the center location with a surrounding area of 10x20 pixels. Also, the information that the eye iris is the darkest area closest to the new point computed can be used , and the eye center location becomes more accurate. Finally, the eye center location of the right eye can be used to compute the eye center location of the left eye because the human face has similarity between the eyes. The authors used this information to search for the left eye within a 10x25 region centered on the vertical dimension with the same horizontal position of the center location of the right eye.

For testing, the algorithm described by the authors used XM2VTS [27] and BioID [28] databases and the results are good enough to be compared with methods that have higher resolution databases. BioID database has around 1500 images of frontal faces with different lightning conditions and backgrounds. Also, this database contains tilted and rotated faces which difficulty's even more the eye detection. The authors used around 600 images from XM2VTS database, and unlike the other database, the images of this database have controlled lighting conditions with an uniform background which facilitates the eye detection. To note that both databases contains persons with glasses and some of the BioID images has people with their eyes closed and various facial expressions.

The results obtained by the authors were divided into: the eye detection and the eye center location. The second one was confirmed visually, observing all the eye center location results of each image. With all of that, for eye detection they achieved accuracy values of 98.2% for the XM2VTS database and 99.1% for BioID database, and for the eye center location the results were 96.7% for XM2VTS database and 98.6% for BioID database.

It can be seen that the algorithm proposed by the authors achieved really good results for both databases and can be compared to other already proposed methods that used higher resolution images. Also, this algorithm works well on non-controlled environment images which proves its robustness.

5.2 Glasses/Sunglasses Detection - ML

ML is a very common area that is used for basically everything in CV and image processing. In order to improve the sunglasses detection, a couple of classifiers were made using two types of ML models that are: Supervised Learning and DL. The big difference between these two type of models is that the supervised models need input features that are extracted from the photos in order to train the network, while the DL ones receives the image and they do by itself the feature extraction.

The work shown below is divided in 2 different solutions.

For the first one, the classifiers were used to detect glasses, instead of sunglasses, and then was used “low-level” methods to decide if the glasses detected are or not sunglasses.

Secondly, a new database was “created”, merging some databases that already existed choosing by hand the sunglasses and no sunglasses photos, to allow the classifiers to detect sunglasses directly. As you will see, the results were far better than the previous solution.

5.2.1 Supervised Learning - Support Vector Machine and Neural Network (NN)

Supervised Learning is a ML approach that uses labeled data. In this case, as it will be shown below, the dataset used has one class that indicates if the person in the photo is wearing glasses or not. Also, Supervised Learning needs as input a set of features. Normally, these features are extracted by "experts" in the specific work area.

The two classifiers, SVM and NN, used to obtain the results shown below were reused from a work. It was only changed the features that were used in that work. The repository link that gives access to the work that describes the work can be found in the Appendices.

Note that, in order to not make this subsections too extensive, only the best results for each classifier will be shown, as well as some comments explaining the results.

Dataset - Supervised Learning

The dataset used for the Supervised Learning Approach was taken from Kaggle [29] and is constituted by 5000 artificial images, where 4500 images are labeled and 500 are not labeled. With this, only the 4500 images with labels were used in the training, cross-validation and test of the classifiers. In Figure 5.1 it is possible to see 2 examples of pictures of this dataset.

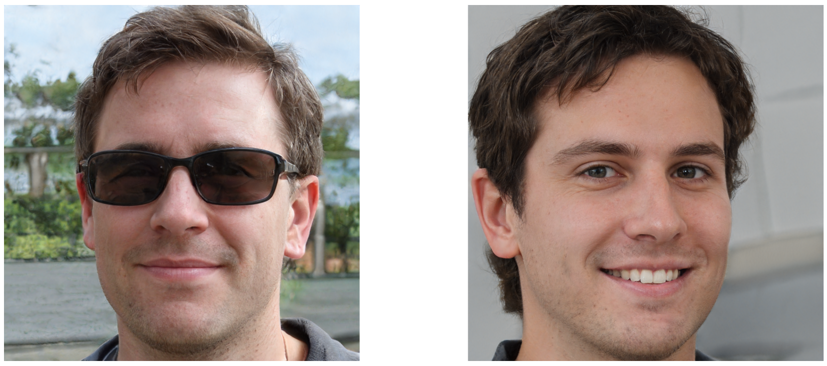


Figure 5.1: Examples from the Kaggle glasses dataset [29]

Feature Extraction for Glasses Detection

After getting the Dataset it is necessary to apply feature extraction in order to get the features needed to train the classifiers. As it will be shown, the results are not as good as expected and, when moving to DL, the improvement is huge, for the simple fact that DL approach has a self-feature extraction. So, the features used for training the classifiers were the following:

- Eye Focus.
- Average pixel intensity of the Eye Region.
- A flag that indicates if edges are present in the Nose Bridge.
- Average Brightness of the Eye Region.

The Eye focus extraction is described in the Appendices because it was a function of the previous software so, it will be not explained again.

Looking to Figure 5.2, the Eye Region mention in the features, is the region of the face width, i.e, x coordinates are between landmark 1 and 17, and the region between the eyebrows and the nose tip, i.e, the y coordinates are between landmarks 20 and the average of 33 and 34. In this case corresponds to the blue rectangle.

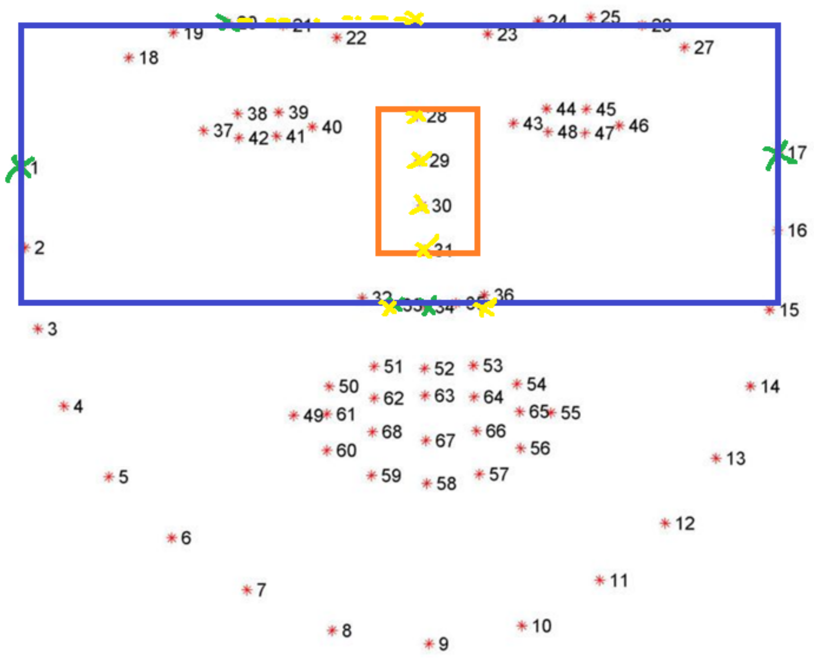


Figure 5.2: `Dlib` 68 Landmarks: Eye Region (blue rectangle and green crosses) and Nose Region (orange rectangle and yellow crosses)

With that the average pixel intensity, as the name said, is the average pixel grayscale values of the eye region.

The average brightness is the average of all bright pixel values of the eye region.

Finally, the flag of the nose bridge [30] is computed using the nose bridge region, in Figure 5.2 is the orange rectangle, i.e, points 28, 29, 30 and 31 (note that 33, 34, and 35 are used too to get the x´s coordinates and 20 to get minimum y coordinate), and then a Gaussian blur and a Edge Detection algorithm, in this case, Canny Edge Detection, are applied giving the Edges of the nose bridge. The flag basically tells if there are edges in the nose bridge, meaning that glasses are present (nose bridge with edges) or not present (nose bridge without edges).

Data preprocessing

In terms of Data preprocessing feature normalization was applied because the features have different ranges of values. For example the average pixel intensity varies between 0 and 255, and the flag of the nose bridge is only 0 or 1. Also, it was seen that the dataset has a class imbalance problem, i.e, there are more examples with glasses, around 63%, and less examples without glasses, around 37%. Then were duplicated the first 1160 images in the data set without glasses in order to get around 50% with and 50% without glasses. Note that there are 4500 images labeled, 2835 (63%) with glasses and 1665 without glasses.

NN

One of the classifiers, as it was said in the beginning of this subsection, was a Classic Fully Connected Neural Network (FCNN). As it is described in the article from the "Foundations of Machine Learning" subject (Appendices), several parameters of the NN were varied, like the number of neurons of the unique hidden layer, the lambda parameter, i.e, the Ridge Regression Regularization, and the alpha parameter, i.e, the learning rate of the NN. First an approach were the features described above were not normalized or balanced was used. As it was expected, the results were very poor, the Confusion matrix and the Performance Metrics are presented in Table 5.1. These results are from the K-fold Cross-Validation approach, where the number of folds is equal to 5.

		Predicted								
		0	1							
Actual	0	0	305	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	0	595	1	0	1	0.661	0.661	0.7959	0.5

Table 5.1: Confusion Matrix and Performance metrics for the NN without feature normalization

As it is possible to observe, the best accuracy was only of 66.11 % for the Test data and they were only False Positives and True Positives. The best parameters chosen for this NN were lambda equal to 0.0001, 25 neurons in the hidden layer and a learning rate of 0.01.

After this, feature normalization was applied and the results are in Table 5.2.

		Predicted								
		0	1							
Actual	0	287	18	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	90	505	0.848	0.941	0.059	0.88	0.965	0.903	0.895

Table 5.2: Confusion Matrix and Performance metrics for the NN with feature normalization

As it can be observed, the results were far better than the ones with non-Normalized data. In this case the accuracy was 88% and the miss-classified images were mostly the False Negatives ones, in this case 90 and “only” 18 false positives. In this case, these results were from a NN with 10 neurons in the hidden layer, with a 0.0001 lambda parameter and a learning rate of 1.

Note that, a fine-tuning was made using the Theta values computed of the best training and cross validation Networks but the results were exactly the same or even worse so they are not shown.

Support Vector Machine

The other classifier used was an SVM, or Support Vector Machine. The same approach as the NN was made, varying the C and Gamma parameters.

Then for the non-Normalized data the results were the following in Table 5.3.

		Predicted								
		0	1							
Actual	0	286	19	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	86	509	0.964	0.7688	0.2311	0.8833	0.8555	0.9065	0.8965

Table 5.3: Confusion Matrix and Performance metrics for the SVM without feature normalization

It is possible to see, that without feature normalization, the results for the SVM are better than the results of the NN, getting a 88.33% of accuracy for a Gamma = 0.01 and C = 50.

However when applying normalization in the SVM data the results are similar to the ones without normalization as it can be seen in Table 5.4.

		Predicted								
		0	1							
Actual	0	287	18	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	88	507	0.9657	0.7653	0.2346	0.8822	0.8521	0.9054	0.8965

Table 5.4: Confusion Matrix and Performance metrics for the SVM with feature normalization

Now the accuracy is 88.22% for a C = 0.5 and Gamma = 1.

To note that the "low-level" approaches, like the HSV comparison between the region below the eyes and the the nose region were only applied to the best classifiers (in this case the DL ones described below), and they were not applied to these results, because the accuracy was already low (maximum of 88.33%) and as the "low-level" methods has flaws this accuracy would only goes down when detecting sunglasses.

In conclusion, with all the results presented in this subsection, it is possible conclude that the Classical Supervised Learning classifiers are not a good choice to detect sunglasses and will not be incorporated in the FotoFaces System.

5.2.2 DL - CNN with 5 Layers and Transfer Learning using VGG16

In this subsection the 2 different DL approaches used to detect glasses and sunglasses will be described. The approaches followed were: transfer learning using the VGG16 architecture and a custom architecture of a CNN with 5 layers. More details about these models are available in the article related to the second project of "Foundations of Machine Learning" (Appendices).

Dataset - DL

In term of dataset, firstly the same one as in the previous classical ML approaches [29] was used to detect if a person is wearing glasses.

As it will be shown, although DL was used, the conjunction of glasses detection with the low-level methods were not good, so the dataset was modified and relabeled to detect sunglasses instead of glasses. Also, some images of a new database [31] were added with sunglasses photos in order to improve the quality and robustness of the dataset and to remove the class imbalance problem, because the sunglasses photos were far less than the non-sunglasses ones. In Figure 5.3 it is possible to observe the added examples of the new dataset to the previous one.

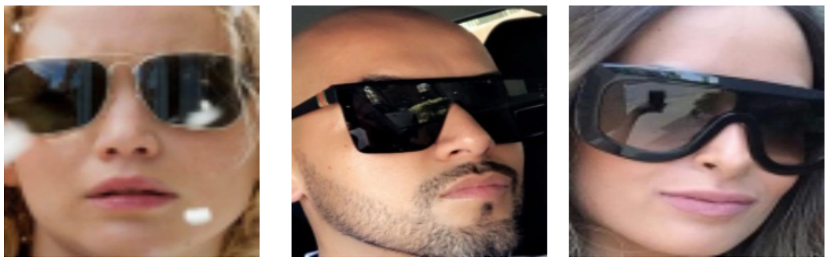


Figure 5.3: New Sunglasses Data Set Examples

As it can be seen, these images are not controlled at all and the faces in them are tilted, rotated and with different expressions, which is the opposite with the photos in the previous dataset where the persons were looking forward with a “controlled” lightning conditions. These new photos add some robustness to the dataset.

Finally, as online data augmentation is made, the division of the data was not made strictly like in the previous one. Most of the images are used to train the classifier, in this case 5675 images, where 1630 are the sunglasses ones and the rest are the images from the previous database with no sunglasses (glasses and no glasses). For validation 435 images (190 with no sunglasses and 245 with sunglasses) are used. Finally, for test proposes 106 images are used where 95 have no sunglasses and 11 with sunglasses. To note that the imbalance problem in the training phase does not exist because, with data augmentation, there are in total 85125 images (training occurs during 15 epochs) from which 24450 images have sunglasses and with so much data the imbalance problem is removed. Also, for the test phase the low number of images with sunglasses is not relevant because a “final” test database was made with all types of profile pictures with and without sunglasses with 114 images. Half of these pictures are from family and friends of myself, in this case, the same person appears up to 3 times with glasses, with sunglasses and without glasses, and the other ones where taken from wikipedia MUG dataset. This new test dataset allows to prove that the model works with photos that are not from the training/validation dataset, i.e., the model works for all types of photos.

In Figure 5.4 there are some examples of the “final” test database.

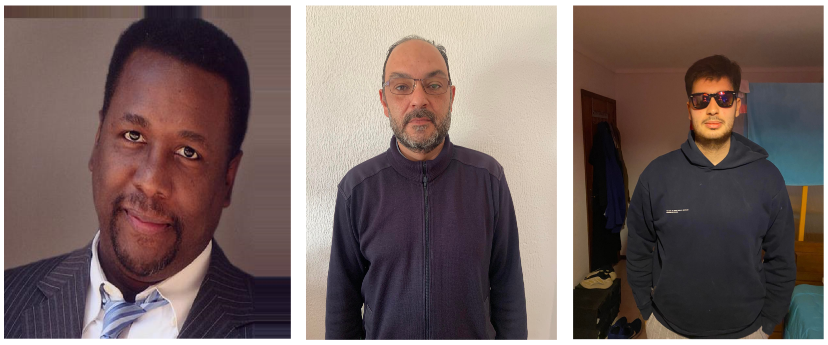


Figure 5.4: Final Test Data set Examples

Sunglasses Detection - CNN with 5 layers and VGG16 model combined with Low level methods

As it is described in the article of Project 2 (Appendices), a CNN with 5 layers was created to train the model to detect glasses with the database from Kaggle [29] and it was used a transfer learning approach with the VGG16 architecture for the same purpose.

In short, the CNN has 5 convolutional layers with a FCNN in the end of the layers. Also, a callback function and dropout layers (one for each convolutional layer) were added in order to prevent overfitting of the training data and vanishing gradient problem. The CNN convolutional layers have 3x3 Padding, stride = 1, a ReLU activation function and Max Polling 2x2. In the end of the FCNN a sigmoid function is used because the classification problem is binary. The Dropout layers have different rates, 0.2 for the first 3 and 0.5 for the final 2. Finally, the total number of parameters of this CNN network is 206 721 where only 320 are not trainable and the network was trained for 15 epochs, in this case after the seventh epoch the accuracy and cross-validation errors were stabilized. To note that these parameters and rates are explained and described with more detail in the respective article to not overextend this subsection.

The VGG16 architecture, as the name suggests, is composed by 16 layers (with weights) with around 138 million parameters. A global average pooling layer, a dropout layer (rate = 0.2) to prevent overfitting, a dense layer with 513 trainable parameters and a sigmoid activation function because the classification problem is binary were added to this architecture. Therefore, the total parameters of this VGG16 model are 14 715 201, where in the first phase only the 513 dense layer parameters are trainable and for the fine-tuning phase there were unfreeze 10 top layers with a total of 13 569 793 trainable parameters. As in the previous CNN model, more details can be found in the article.

With all of that said, let´s have a look into the glass detection results for these 2 models that are shown in Tables 5.5 and 5.6.

		Predicted								
		0	1							
Actual	0	86	0	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	1	113	0.991	1	0	0.995	1	0.996	0.996

Table 5.5: Confusion Matrix and Performance metrics for the CNN with 5 Layers with Dropout and Callback Function

		Predicted								
		0	1							
Actual	0	83	3	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	0	113	1	0.965	0.0349	0.985	0.974	0.987	0.983

Table 5.6: Confusion Matrix and Performance metrics for the VGG16 model after Fine-Tuning

As it can be seen, with DL the glasses detection over the same dataset (including the test one) is considerably better than the Classical ML approaches with NN and SVM. Here the accuracy is 99.5% with only one false negative for the CNN and 98.5% accuracy with 3 false positives for the VGG16 model. Of course, all the performance metrics for both models are good as well. This can be easily explained by the simple fact that DL uses self feature extraction, and this is a hard job for a human to extract and give the features to the Classical NN or the SVM.

Despite the glasses detection values, when combined with low-level methods to detect sunglasses, many flaws arrive because these low level methods can be appropriate for some images if there are no reflections in the sunglasses for example but really bad for other images where this reflection is present.

Before going to the low level results, the accuracy values for the “final” test dataset were of 96.7% for the CNN with 5 layers and 97.8% for the VGG16 model. These results are a bit lower than the previous ones on the other test dataset, but as it was said before, this test database is more realistic than the other one and has more sunglasses examples.

The low-level methods used to detect the sunglasses after the glasses detection were the following:

- Compare the HSV color values of the nose region with the HSV of the region below the eyes (Figure 1.3)
- Using a Haar eye cascade, i.e., in case that the eyes are present after the glasses detection means that the person do not have sunglasses
- Compute the Eye Aspect Ratio (EAR)
- Compute the GAZE, i.e., focus of the eyes

Therefore, the results obtained for different low-level methods after using this CNN network for glasses detection over the “final” test dataset. In Table 5.7 it is possible to see these results.

	CNN5	VGG16
HSV	83.30%	83.30%
EAR	52.60%	52.60%
Haar cascade	68.40%	68.30%
GAZE	75.40%	75.40%

Table 5.7: Accuracy Values for Sunglasses Detection in final test dataset for the CNN with 5 layers and the VGG16 model combined with low level methods

As it can be seen, the results are not satisfying, with a maximum accuracy of 83.3% using the HSV comparison method. Of course, this happens because the low level methods are not robust enough, and for uncontrolled environment images it is difficult to find some method that can fulfill all the possible artifacts that may appear like reflections, occlusions, etc.

Sunglasses Detection - CNN with 5 layers and VGG16 model for direct classification

In order to improve the previous results, the same networks were used but now directly over the modified database for sunglasses detection, i.e., classifying directly if a person is wearing or not sunglasses, instead of using low level methods for this job.

Therefore, using the same final test dataset, the results for the CNN with 5 layers and the VGG16 models can be observed in Tables 5.8 and 5.9.

		Predicted								
		0	1							
Actual	0	0	73	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	0	41	1	0	1	0.36	0.359	0.529	0.5

Table 5.8: Confusion Matrix and Accuracy Values for Sunglasses Detection in final test dataset for the CNN with 5 layers

		Predicted								
		0	1							
Actual	0	59	14	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	8	33	0.805	0.808	0.192	0.807	0.702	0.75	0.807

Table 5.9: Confusion Matrix and Performance Metrics for Sunglasses Detection in final test dataset for the VGG16 model

As it can be seen, the results are really bad for the CNN with 5 layers over the final test database for sunglasses detection, where all the predictions says that the person have sunglasses, i.e., 0 true negative in the confusion matrix with only 36% of accuracy. On the other hand, the VGG16 has similar results to the low-level methods, in this case 81% of accuracy with 8 false negative and 14 false positive values.

These results, as it will be seen, can easily be improved observing the so called attention maps and adding some data preprocessing before jumping into the training phase of the model.

5.2.3 Attention Maps over VGG16 model

A way to know what the model is considering in order to classify an image, i.e., what regions of the image are important for the inference, is using attention maps. Basically, an attention map in DL, is a grid of numbers that indicates which regions of an image (2D space) are important for some given task. It can be represented by an heatmap, where the red regions (warmer regions) represents the important parts of that images for classification. To note that in some literature these attention maps are referred as activation maps.

With all of that said, the attention maps were generated for final test database used to test the VGG16 model. It was necessary to use a "handmade" VGG16 [32] because with the one available on Keras (built-in function), it is not possible to access the convolutional layers, in this case the last convolutional layer, when adding new layers to it.

Also, as it is described in [16], data preprocessing can have significant impact over the CNN models. The first preprocessing method applied was the verification of the Image Quality. In this case, BRISQUE [33] and Koncept224 [2] were used to verify if the image is minimally good to be used as input to the VGG16 network. In this case, a threshold of 50 was used for the BRISQUE model (0 best image quality value and 100 worst value) and for Koncept224 a threshold of 15 was used (100 best image quality value and 0 worst quality value). To note that these threshold values were based on the quality values for the training, validation and test datasets that were computed and then analysed manually.

Also, a face detector from `Dlib` was applied (described in the Appendices) in order to verify if there are any faces in the photo and a rotation of the image was made based on the face (if present).

Before showing some of the examples of the attention maps for the "handmade" VGG16 model, the training results with the image quality preprocessing BRISQUE and Koncept224, combined with the face detector and the rotation of the face, are presented on Tables 5.10 and 5.11 respectively.

		Predicted								
		0	1							
Actual	0	66	4	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	12	21	0.636	0.943	0.0571	0.845	0.84	0.724	0.79

Table 5.10: Confusion Matrix and Performance Metrics for Sunglasses Detection for the "handmade" VGG16 model with BRISQUE

As it can be seen, the results for the VGG16 with BRISQUE, 84,5% accuracy, are better than the results with Koncept224, 76.9% accuracy. The big difference is that BRISQUE rejected more images to train, validate and test the network, and therefore as the data has more "quality", then it has less false positive samples that Koncept224. In this case, the

		Predicted								
		0	1							
Actual	0	60	13	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	12	23	0.657	0.822	0.178	0.769	0.639	0.648	0.74

Table 5.11: Confusion Matrix and Performance Metrics for Sunglasses Detection for the "handmade" VGG16 model with Koncept224

face detector and the rotation has no significant impact because they are the same for each quality model. To note that in [2] are compared the 2 models used to compute the quality and, Koncept224 has better performance scores than BRISQUE, but in this case and with the threshold values chosen BRISQUE had a better preprocessing impact than Koncept224 over the network operation.

In term of attention maps, some results from each model can be seen Figures 5.5 and 5.6.

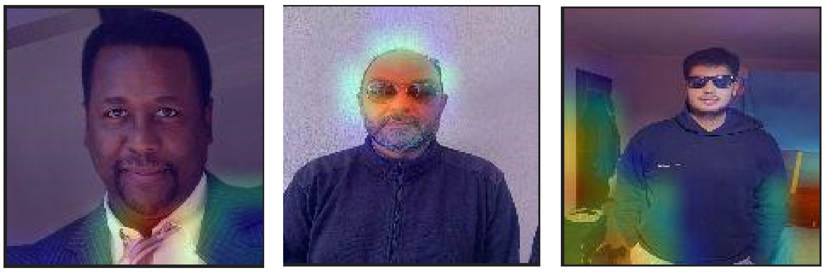


Figure 5.5: Attention maps for the "handmade" VGG16 with BRISQUE

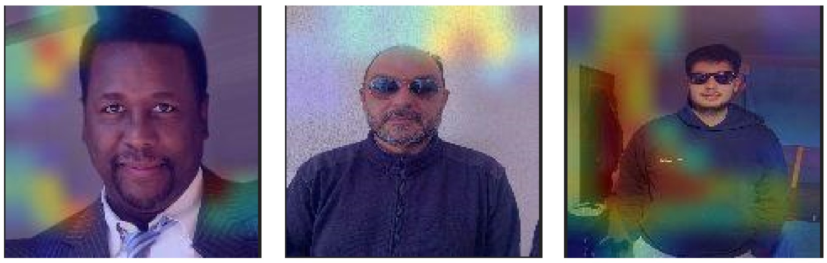


Figure 5.6: Attention maps for the "handmade" VGG16 with Koncept224

Also, the averaging of the attention maps was made for the sunglasses photos and for the non-sunglasses photos in order to observe with more detail what regions is the CNN considering to classify each photo. This averaging can be seen in Figures 5.7 and 5.8, where each averaging of the attention maps is overlaid over an example of the respective label (sunglasses/no sunglasses) for each image quality model.

It can be observed that for the sunglasses images the CNN with BRISQUE is "almost" considering the right areas of the image to classify them, i.e., the eye region where sunglasses can be present, while Koncept224 is using the background noise to detect if there are sunglasses present or not in the image. For both models, in the no sunglasses images the warmer regions, if they exist, are in the borders of the image, i.e., the background, because there are no sunglasses in it. Although these results are not bad at all, this averaging of the attention maps over the original images can not be used to draw conclusions because the sunglasses are not located in the same regions of each image.

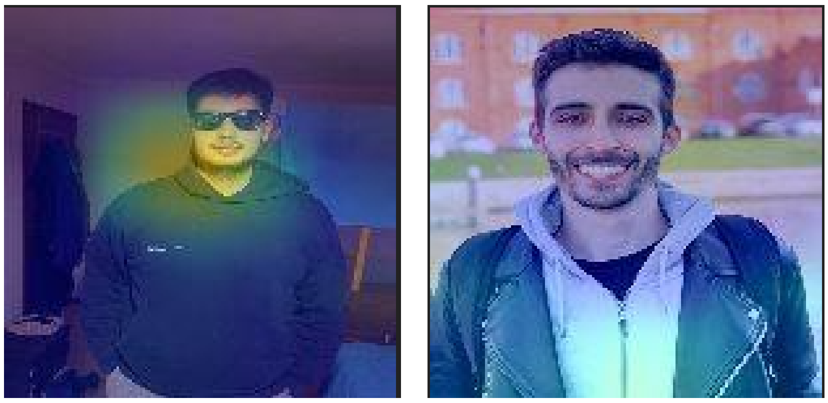


Figure 5.7: Averaging of Attention maps for the “handmade” VGG16 with BRISQUE

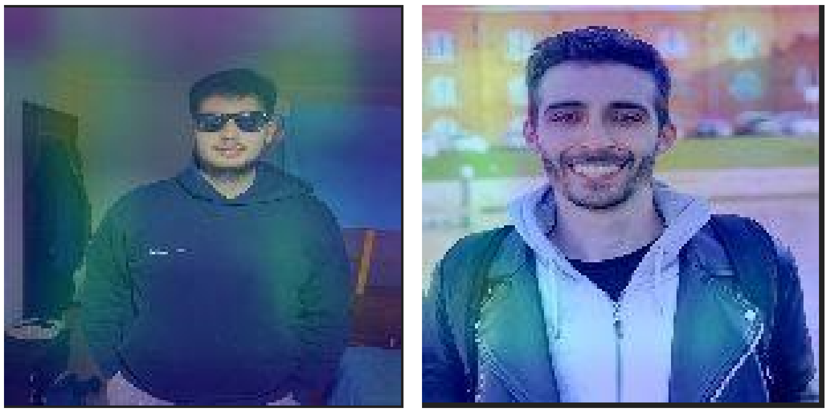


Figure 5.8: Averaging of Attention maps for the “handmade” VGG16 with Koncept224

Also, as the accuracy values are still low, a cropping of the face region of each image to the previous data preprocessing methods was applied. With that, the sunglasses are now located in the same region in every image. In this case, the landmarks that define the face limits of each person were used, which are points 1 and 17 for the x limits and 20 and 9 for the y limits. These face region can be observed in Figure 5.9 as the red rectangle.

The confusion metrics and the performance metrics for each model with cropping are present in Tables 5.12 and 5.13 and the averaging of the attention maps in Figures 5.10 5.11.

		Predicted								
		0	1							
Actual	0	70	0	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	4	29	0.879	1	0	0.961	1	0.935	0.939

Table 5.12: Confusion Matrix and Performance Metrics for Sunglasses Detection for the “handmade” VGG16 model with BRISQUE and face cropping

		Predicted								
		0	1							
Actual	0	73	0	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	7	28	0.8	1	0	0.935	1	0.889	0.9

Table 5.13: Confusion Matrix and Performance Metrics for Sunglasses Detection for the “handmade” VGG16 model with Koncept224 and face cropping

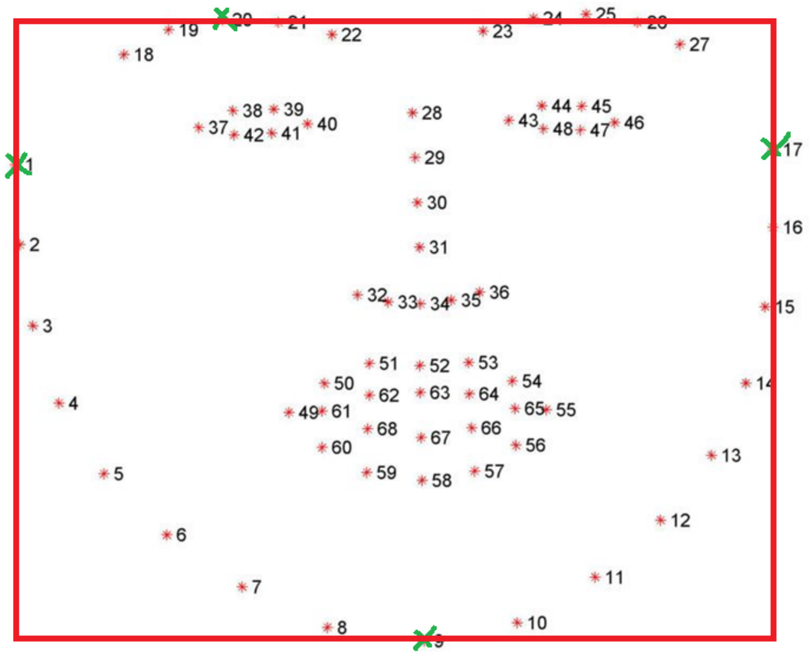


Figure 5.9: `Dlib` 68 landmarks: Face Region (red rectangle)

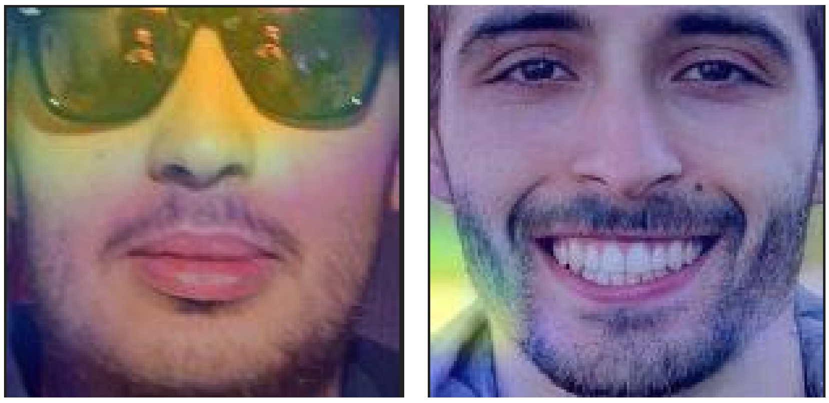


Figure 5.10: Averaging of Attention maps for the "handmade" VGG16 with BRISQUE and face cropping

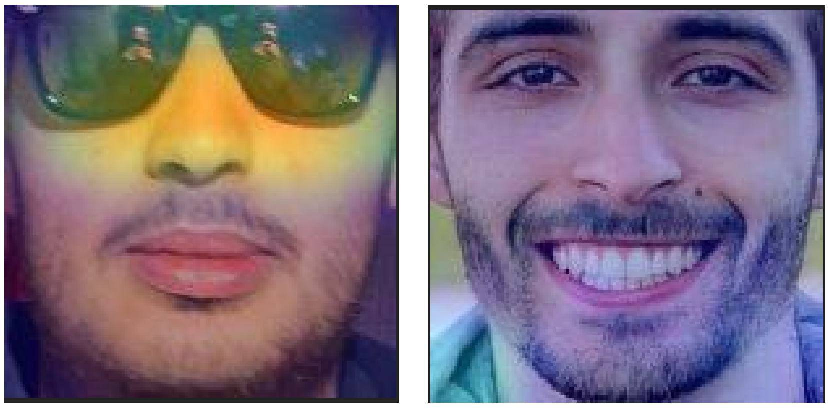


Figure 5.11: Averaging of Attention maps for the "handmade" VGG16 with Koncept224 and face cropping

It can be seen that the accuracy values of each model has improved significantly with this cropping, 11,6% for the VGG16 with BRISQUE and 16,6% for the VGG16 with Koncept224. Also, the number of false positive examples has disappeared for both models because as there is no background noise and the number of false negatives examples has been reduced as well. In terms of the attention maps, it can be seen that for the sunglasses photos, the models are considering the sunglasses region with a little bit of the nose region to detect sunglasses. This proves that the cropping was important and the background influenced the way the models were detecting sunglasses, as it was shown in [16]. For the no sunglasses photos, the same happens as before, where there are a few warmer regions on the border of the images if they exist.

Finally, another cropping was applied to the previous one, but now, only considering the eye region merntion before. In this case, observing Figure 5.2 points 1 and 17 for the x limits of the image and points 20 and 34 for the y limits were used (blue rectangle).

The confusion matrix and the performance metrics of each model can be observed in Tables 5.14 and 5.15 and the respective averaging of the attention maps in Figures 5.13 and 5.13.

		Predicted								
		0	1							
Actual	0	70	0	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	2	31	0.939	1	0	0.981	1	0.969	0.97

Table 5.14: Confusion Matrix and Performance Metrics for Sunglasses Detection for the “handmade” VGG16 model with BRISQUE and eye region cropping

		Predicted								
		0	1							
Actual	0	73	0	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	4	31	0.886	1	0	0.963	1	0.94	0.943

Table 5.15: Confusion Matrix and Performance Metrics for Sunglasses Detection for the “handmade” VGG16 model with Koncept224 and eye region cropping

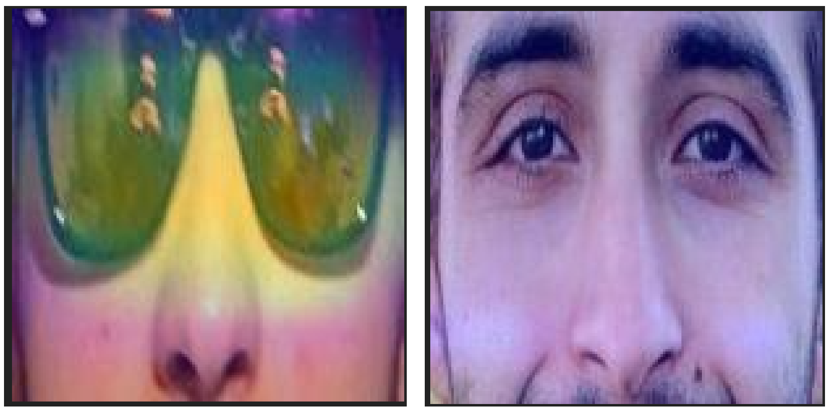


Figure 5.12: Averaging of Attention maps for the “handmade” VGG16 with BRISQUE and eye region cropping

As it can be seen the results improved by around 2% for each model, in this case VGG16 with BRISQUE has now 98.1% of accuracy and VGG16 with Koncept224 has 96.3%. The

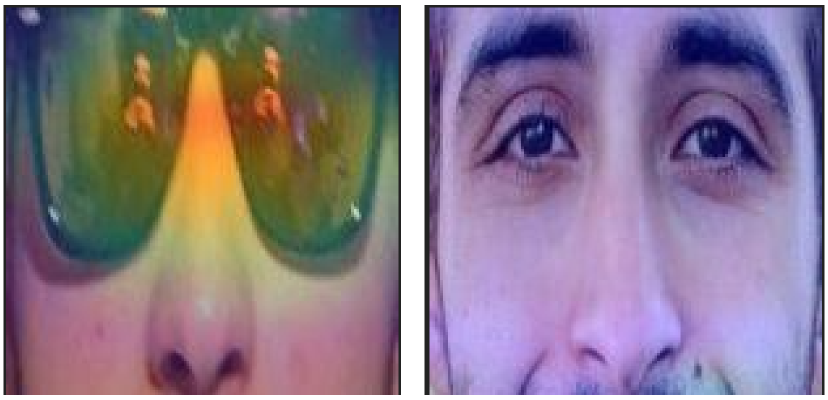


Figure 5.13: Averaging of Attention maps for the "handmade" VGG16 with Koncept224 and eye region cropping

VGG16 with BRISQUE missclassified only 2 images and the VGG16 with Koncept224 4 images, both false negative ones. Also, the attention maps focus more on the sunglasses regions for both of the models and do not use the nose region like the previous ones. As it was said before, the VGG16 with BRISQUE got better results because it discards more images with the threshold chosen and the quality of the images is better to classify the sunglasses.

As an extra attempt, the model with image smoothing over the cropped images was trained, in this case using a 5x5 Gaussian Kernel, but unfortunately the accuracy value was only of 92% so the results will not be shown here. It can be concluded that in this case the Gaussian Blur did not affected positively the sunglasses detection.

5.3 Comparison with the previous method

As explained in the Appendices, the previous method implemented in the FotoFaces to detect sunglasses was based on the comparison between the HSV color values of the nose region and the region below the eyes. Although this method looks simple and good, the results over the final sunglasses test dataset were only 79% of accuracy. This means that the transfer learning model VGG16 with the eye region cropping has improved the accuracy by 19.1%.

In Table 5.16 it is possible to observe the evolution of accuracy values between the old (worst) and the best method of sunglasses detection. To note that the VGG16 with Koncept224 results are not present since they were always worst than the VGG16 with BRISQUE.

	Accuracy
HSV	79.00%
VGG16 and HSV	83.30%
CNN5 aand HSV	83.30%
VGG16 with BRISQUE	85.00%
VGG16 with BRISQUE and face cropping	96.10%
VGG16 with BRISQUE and eye cropping	98.10%

Table 5.16: Accuracy values for the different Sunglasses detection methods

In conclusion, it was shown that using CNN´s, in this case the VGG16 architecture through transfer learning, with a good data centric approach and preprocessing methods, the sunglasses detection becomes an easy and feasible task and can be implemented over any CV system. Also, the improvement of the previous sunglasses detection method was done with success.

CHAPTER 6

Hat detection

In this chapter a DL solution for hat detection similar to the solution proposed in the previous chapter (Sunglasses Detection) is presented, using the same network model as well. Also, a new hat detection database is created merging existing databases and adding some images from others and labeling by hand the images. The creation of this new database was a necessary step because, as far as my research went in the recent hat detection papers, all of them were about hard-hat detection and there was no “general” hat detection dataset available.

6.1 Literature Review

In this subsection there are presented only 2 recent works about hard-hat detection which, as it will be seen, they will be not reused or implemented in the FotoFaces system, since their approaches are model centric, while the approach followed for FotoFaces system was mainly data centric.

Rahul N. Bhadeshiya et al.[34] propose an innovative hard-hat detection algorithm over images and videos dataset from Roboflow using the You Only Look Once (YOLO)v4 network model.

They have used Darknet-53 as the backbone of the YOLOv4, a Path Aggregation Network (PAN) as the neck of YOLOv4 and YOLOv3 as the head of YOLOv4 in order to achieve faster detection results.

The images from the Hard Hat Roboflow dataset were rescaled to 64x64 pixels and transformed into 3x64x64 tensors. For implementation purposes the authors used IoU limit equal to 0.5.

In terms of experimental results, the authors achieved 99% of accuracy for hard-hat detection in images and 98% of accuracy for videos.

Although the results obtained by the authors are good, the work is not compared with any hard-hat detection works and this detection is tested only over the same dataset which do not guarantees a good functionality over others existing datasets.

Nikolav Filatov, Natalia Maltseva Aleksandr Bakhshiev et al. [35] present an automatic hard-hat detection algorithm that uses Deep Neural Network (DNN)s with High Interference speed. The dataset used to train, validate, and test the network was constituted by 1478 images from various scenes.

The authors used an end-to-end NN with one stage object detection. First, they tested 3 different networks: SqueezeDet, Tiny-YOLO and RetinaNet to see which interference time was the lowest and, in this case, the SqueezeDet won. To evaluate the SqueezeDet the authors used 3 measures: recall, precision and F1 score. The results obtained were the following: 0.55 for precision, 0.91 recall value and 0.69 for F1 score.

In order to improve the network the authors added image preprocessing, in this case, resizing the images to 1280x720 resolution and normalizing them, and for some categories of classification (no hard-hat detection) they added an additional classifier. In this case, MobileNet was added with the lowest inference speed for this additional classification. With that, the evaluation values improved to 0.64 of precision, 0.91 of recall and 0.75 of F1 score.

This work shows good results and low inference speed of the network model, but like the previous one, no comparison with other hard-hat works have been done.

6.2 Hat Detection - DL

As it was described in the precious section, the works that exist on hat detection are in the area of civil construction, more specifically detecting if a person is wearing a safety helmet, or more commonly mention the hard-hat. Also, it was researched if there were any datasets to be used in hat detection and a dataset with around 800 images of 4 types of hats was found, namely hijab, medical cap, turbans and winter hats with approximately 200 images for each type of hat.

With that, it was necessary to follow a data centric approach, constructing a hat detection dataset which could cover most of hat examples or headpieces that exist. This new dataset was made using several images from other datasets [29], [36], [37], [38] and [39]. It is constituted by around 9530 images. Also, a split of the dataset was made for training, cross-validation and test purposes. In this case, the training set contains 8287 images where 1150 are images of people wearing hats/headpieces and 7137 images of people not wearing anything. The validation set contains 700 images where 307 are images with hats/headpieces and the rest 393 images without hats/headpieces. Finally, the test set contains 543 images with 237 with hats/headpieces and 306 without.

Some examples of this new dataset can be seen in Figure 6.1.

The model used for hat detection was the same used for sunglasses detection, i.e., the transfer learning VGG16 model. With that, some preprocessing methods were added to the input images of the network as before, more specifically, verify the quality of the image using BRISQUE and Koncept224 models, and in case that the the image has the minimum quality, give the original image as input to the network with a resized to 150x150 resolution or apply a cropped version of the image with the face region and the above face region (forehead and head regions), in this case where the hats normally are located. In this case, the different

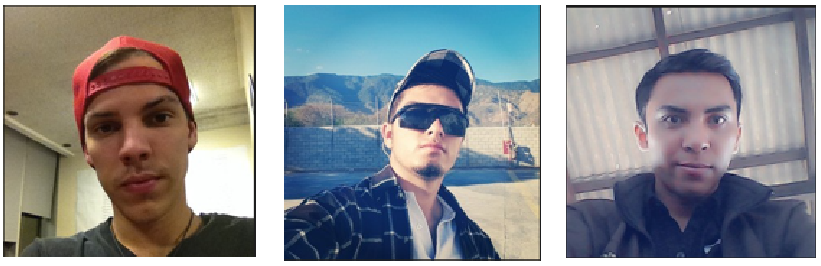


Figure 6.1: Hat Detection Dataset Examples

regions can be observed in Figure 6.2. The face with forehead and head region is represented by the red rectangle, while the forehead and head region is represented by the blue rectangle. The face with forehead and head region is delimited by the landmarks 1 and 17 for the x limits and by the landmark 9 for the upper y limit and for the lower one, is computed 75% of the distance between the landmark 1 and 17 and then is subtracted to the y average point between landmark 20 and 30. The same is applied for the forehead and head region but the upper y limit is the landmark 20 (eyebrow). Several visual tests (including training and testing the model) were made and this 75% value was the best one of them all in terms of performance metrics.

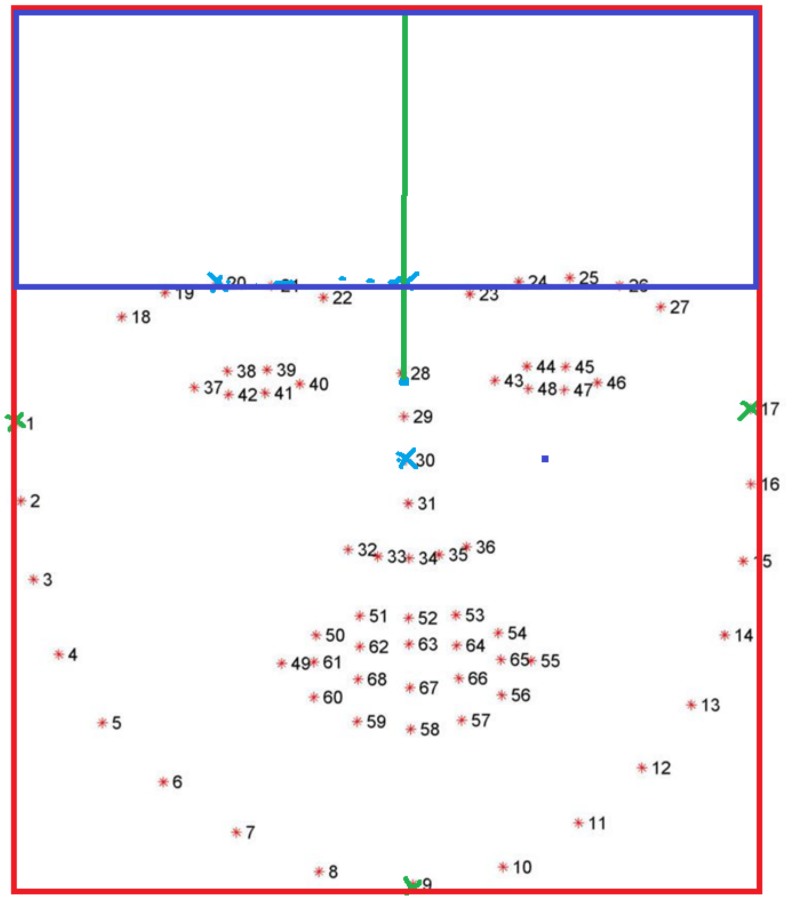


Figure 6.2: `Dlib` 68 landmarks: Face with forehead and head Region (red rectangle) and Forehead and Head Region (blue rectangle)

6.2.1 VGG16 model - Results and Attention Maps

As it was said before the model was trained using some preprocessing methods. First of all let´s have a look into the results for the model using the BRISQUE and Koncept224 to verify the input image quality and using the original image as input. These results can be seen in Tables 6.1 and 6.2.

		Predicted								
		0	1							
Actual	0	226	6	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	38	120	0.759	0.974	0.0259	0.887	0.952	0.845	0.867

Table 6.1: Confusion Matrix and Performance Metrics for Hats Detection using the "handmade" VGG16 model with BRISQUE

Observing the confusion matrix for both image quality models, it can be seen that the VGG16 with BRISQUE has a better impact over the input training images than the VGG16 with Koncept224 with less missclassified examples (false positives and false negatives). Also, it can be seen that both models have just a few false positive examples, i.e., people not wearing hats classified has wearing hats, but they have a lot of false negatives, which means

		Predicted								
		0	1							
Actual	0	235	3	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	50	111	0.689	0.987	0.0126	0.867	0.974	0.807	0.838

Table 6.2: Confusion Matrix and Performance Metrics for Hats Detection using the "handmade" VGG16 model with Koncept224

that the models can not detect all types of hats. Nevertheless, both models, perform "okay" having 88,7% and 86.7% of accuracy values respectively. Noting that, for Hat detection it is considered a good accuracy a value of 95% or higher.

To understand better the way the models were classifying these hats, the average Attention Maps of each model that are represented on Figures 6.3 and 6.4.

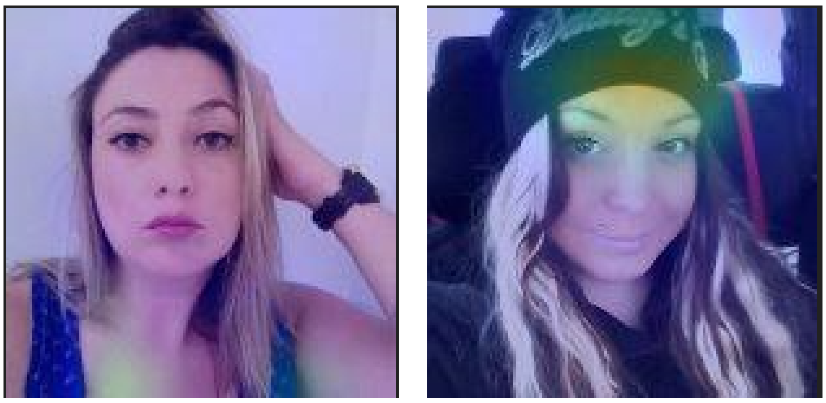


Figure 6.3: Averaging of Attention Maps for Hats Detection using the "handmade" VGG16 model with BRISQUE

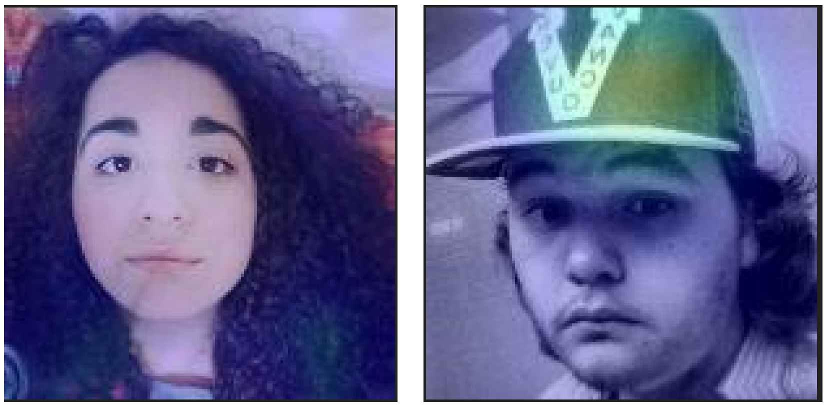


Figure 6.4: Averaging of Attention Maps for Hats Detection using the "handmade" VGG16 model with Koncept224

As it can be seen, both of the averaging of Attention maps are "good" because the activation to decide if there are or not hats is over the face/head of the person with an hat (left picture) and in the picture with no hat the activations are in the background. Although, when looking through each photo individually, it was seen that in most of the cases the activation was not over the hat or headpiece but instead, over the face as well or some background noise. This can lead to a bad performance over another dataset that contains images from other environments. Also, some of the non-hat photos got activations in the background noise as well. Some of these examples are illustrated over Figures 6.5 and 6.6. Again, this averaging over the original images can not be used to conclude that the network is working properly, because the hats are not in the same location for each every image.

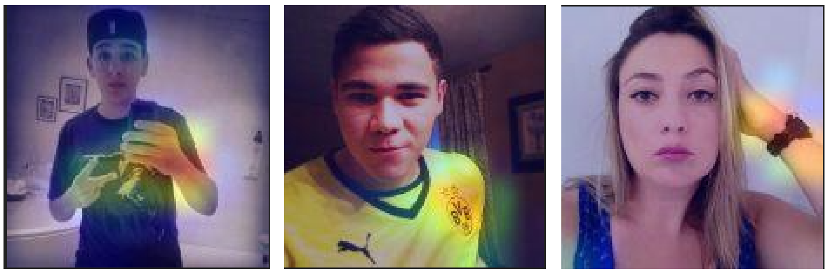


Figure 6.5: Examples of Bad Attention Maps for Hats Detection using the "handmade" VGG16 model with BRISQUE

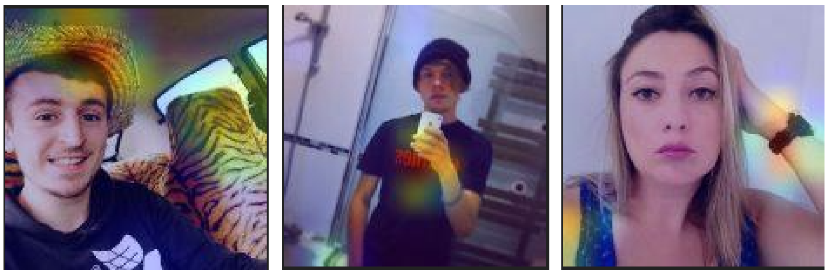


Figure 6.6: Examples of Bad Attention Maps for Hats Detection using the "handmade" VGG16 model with Koncept224

To correct these activation maps and centering the hats location, another preprocessing method over the input training images was added. In this case, a crop over the face region with forehead and head region was added (Figure 6.2 red rectangle).

With that, the models were trained again and the results can be seen in Tables 6.3 and 6.4.

		Predicted								
		0	1							
Actual	0	203	9	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	25	86	0.774	0.957	0.0425	0.895	0.905	0.835	0.866

Table 6.3: Confusion Matrix and Performance Metrics for Hats Detection using the "handmade" VGG16 model with BRISQUE and cropping

		Predicted								
		0	1							
Actual	0	195	17	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	19	95	0.833	0.92	0.08	0.89	0.848	0.841	0.88

Table 6.4: Confusion Matrix and Performance Metrics for Hats Detection using the "handmade" VGG16 model with Koncept224 and cropping

As it can be seen, the results with cropping in terms of accuracy, are slightly better than the ones without cropping with 89.5% and 89% of accuracy for VGG16 with BRISQUE and

Koncept224. Also, when looking to the confusion matrix it can be seen that now, the number of false positives and false negatives is more balanced, i.e, there are a lot less false negatives but the number of false positives has increased. This could mean that the models now can classify more types of hats, but in some cases people with no hat are being classified as wearing a hat.

Looking to the Averaging of the Attention Maps over Figures 6.7 and 6.8 and to each attention map individually, it is possible to see that the model is "detecting" the hats instead of background noise or the face of the person in the photo. Some examples of attention maps with cropping are shown Figures 6.9 and 6.10. To note that in these examples it is possible to observe that, in some photos, regular hair is considered as a hat too.

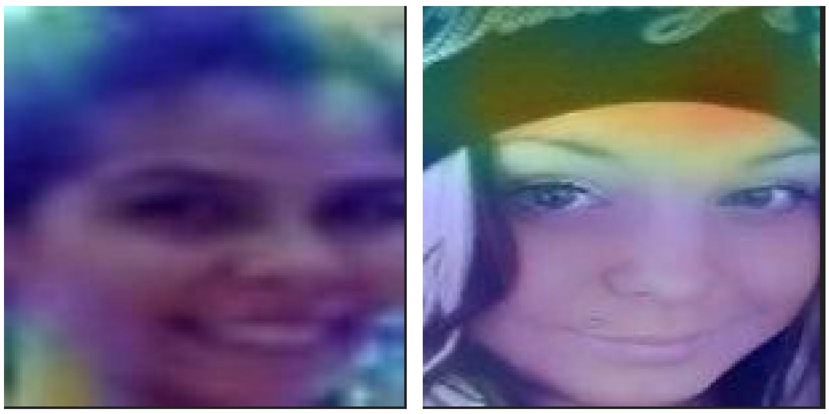


Figure 6.7: Averaging of Attention Maps for Hats Detection using the "handmade" VGG16 model with BRISQUE and cropping

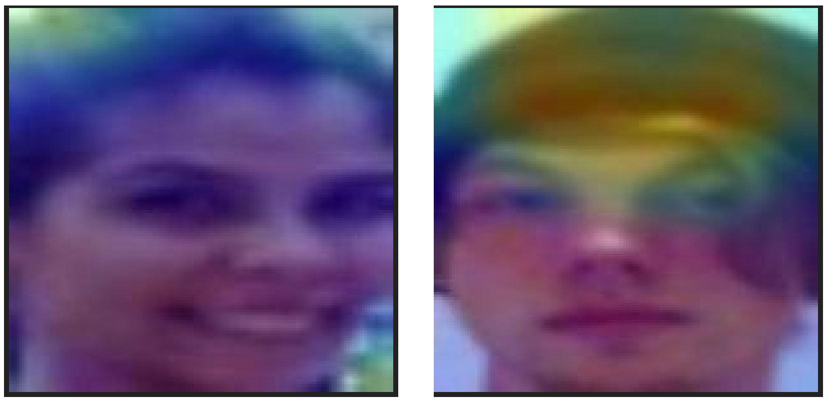


Figure 6.8: Averaging of Attention Maps for Hats Detection using the "handmade" VGG16 model with Koncept224 and cropping

Finally, just like in the sunglasses detection, a focused cropping over only the forehead and head region was made, in this case the region above the eyebrows and the 75% value using the landmarks of the eyebrow and the nose mention before. These two regions (forehead and head) are shown in Figure 6.2 (blue rectangle).

With that, the results of hat detection for VGG16 with BRISQUE and VGG16 with Koncept224 can be seen in Tables 6.5 and 6.6 and the respective averaging of Attention maps in Figures 6.11 and 6.12.

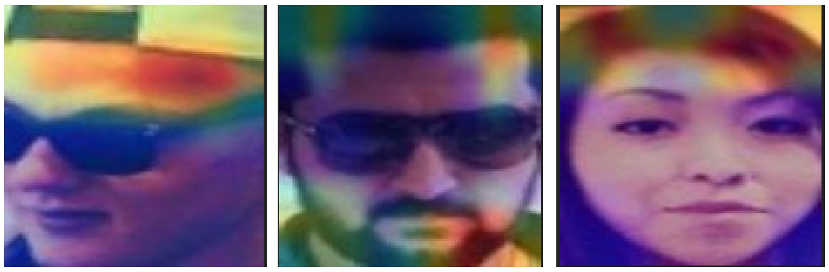


Figure 6.9: Examples of Bad Attention Maps for Hats Detection using the "handmade" VGG16 model with BRISQUE and cropping

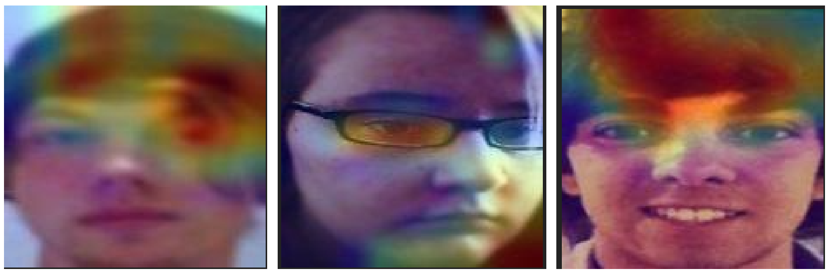


Figure 6.10: Examples of Bad Attention Maps for Hats Detection using the "handmade" VGG16 model with Koncept224 and cropping

		Predicted								
		0	1							
Actual	0	211	1	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	37	75	0.67	0.995	0.00471	0.882	0.987	0.798	0.832

Table 6.5: Confusion Matrix and Performance Metrics for Hats Detection using the "handmade" VGG16 model with BRISQUE and head cropping

		Predicted								
		0	1							
Actual	0	209	3	Recall	Specificity	FPR	Accuracy	Precision	F1	Balanced Accuracy
	1	33	82	0.713	0.986	0.0141	0.89	0.965	0.82	0.849

Table 6.6: Confusion Matrix and Performance Metrics for Hats Detection using the "handmade" VGG16 model with Koncept224 and head cropping

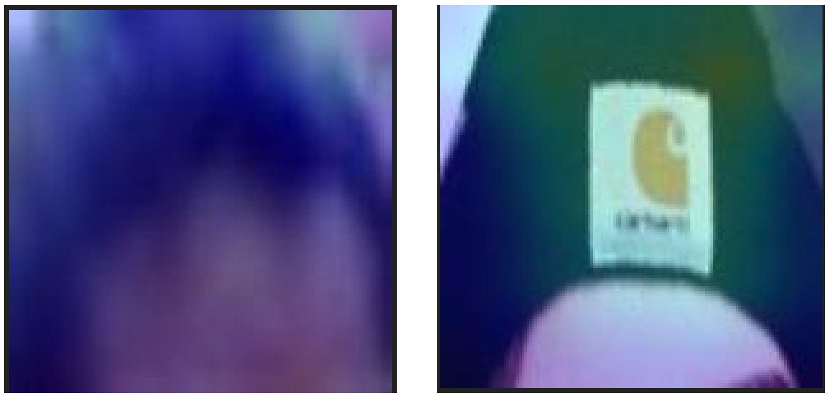


Figure 6.11: Averaging of Attention Maps for Hats Detection using the "handmade" VGG16 model with BRISQUE and head cropping

As it can be seen, the results were slightly worse than the results for the previous cropping (with the face region) with 88.2% and 89% accuracy values for the VGG16 with BRISQUE

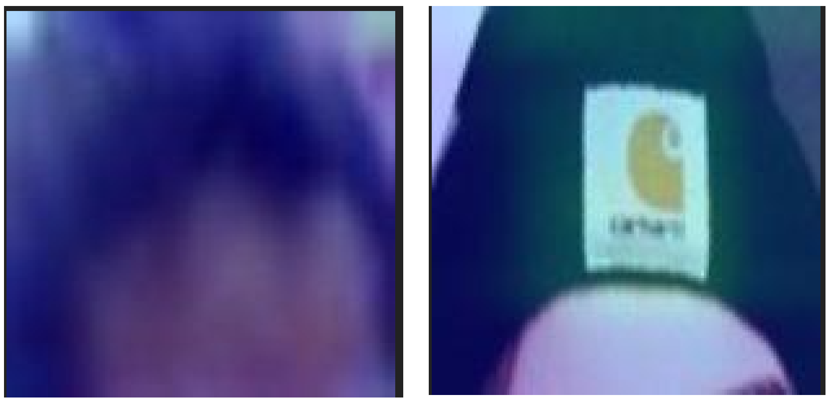


Figure 6.12: Averaging of Attention Maps for Hats Detection using the “handmade” VGG16 model with Koncept224 and head cropping

and Koncept224. Also, as in the first model, the number of false positives is small and the number of false negatives has increased again, which means that a lot of hats is not detected. Although the average attention maps are good, when looking at some individual attention maps of the test images it is possible to observe that this model has flaws. These examples are shown in Figures 6.13 and 6.14.

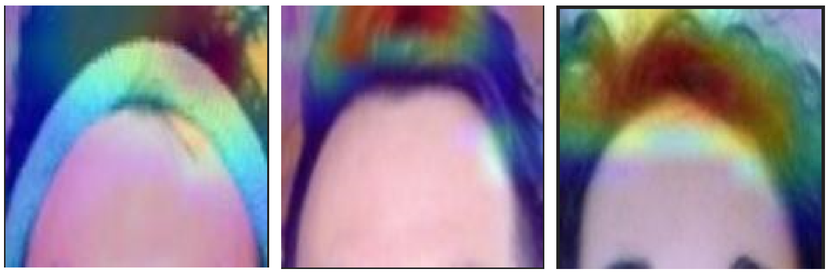


Figure 6.13: Examples of Bad Attention Maps for Hats Detection using the “handmade” VGG16 model with BRISQUE and head cropping

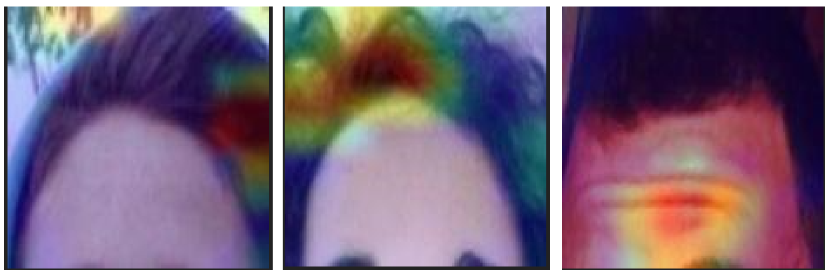


Figure 6.14: Examples of Bad Attention Maps for Hats Detection using the “handmade” VGG16 model with Koncept224 and head cropping

Next, a k-fold cross validation method with 5 folds was applied to the head cropping preprocessing model. Again the results were similar to the previous ones with an average of 88% of accuracy. Also, there was no significant difference in the attention maps so these results will not be shown to not overextend this chapter.

Finally, the 3 trained models (no crop, face region with hair region crop and forehead and hair region crop) were tested, over the sunglasses “final” test dataset, which only has 4 images with hats. Despite that, the discrepancy between the results was huge and was a way of choosing the model to be implemented into the FotoFaces system. The accuracy for the no-cropped model was only of 70%, which is quite low and did have 1 false negative example comparing to the 23 negative examples. The face and hair region cropped model obtained a 80% of accuracy with no false negative examples but, just like the previous one, a lot of false positive examples, in this case 21. The forehead and hair region model performed well with 96% of accuracy with 1 false negative example and only 3 false positive examples. To note that it can not be said that low number of the false negative examples means that the model performs well in recognizing different types of hat because there are only 4 images with hats in this test dataset. With all of that, the model chosen to be implemented on FotoFaces system was the one with the forehead and hair region cropping.

As a final view of this chapter, the preprocessing methods, more specifically cropping, had great impact over the hat detection results. In this case, this impact caused by cropping was mainly over the attention maps and not so much in the performance metrics values.

CHAPTER 7

Image Background Analysis

Background Analysis has been a commonly explored topic over the last few years. The main objective of this chapter is to present some algorithms that can be used to compute the tilt angle of the background on the image submitted by the user after the face rotation has been applied.

As it will be seen in this chapter, some basic low level methods for helping the tilt angle correction were explored and 2 of the Literature review methods were implemented/reproduced.

7.1 Literature Review

In this subsection 5 different works are presented, where 3 are from the same authors, that can be used to compute tilt angle correction. Although there are more works related to this topic, they were not included into this review since the most recent work from the 3 semantic line detection works is the best one, to the best of my knowledge. Also, two of the works are just simple low level approaches, while the other 3 are a constant evolution of a semantic line detector network.

Jaroslav Matej et al. [40] presents a forestry machines tilt angle algorithm using some image processing "basic" methods combined with the information extracted by some tilt sensors.

Initially, the author proposes a method consisting of applying Sobel operators for edge detection followed by Hough Transform for line detection. After that, a line quality analysis is made using each line length information and completeness. Finally, the tilt angle of the best lines is computed. Although this algorithm seems relatively simple and suitable for the FotoFaces, it takes more than one minute to compute the tilt angle, i.e, it is too slow and Hough transform can not be applied to detect omnidirectional lines.

With that, the author proposes a second and simpler approach that fulfills all the objectives, which is faster and more accurate than the previous one. This one uses the Sobel operator for edge detection. Then, the edge image columns and rows are used to compute the sum of line pixels for each line, i.e, for each column each pixel row is analyzed and in case that it is an edge (white pixel) and the next row pixels are edges too, i.e, part of a line where the line ends if a certain limit of row pixels are not edges, then the number of edges for that column are added to the final line results for that image. With this, a rotation to the image of one degree is made and the sum of the lines detected is computed. In this case, the author computes the values between -15º and 15º degrees values of the original value and then the biggest line sum of the image is used to tell what the tilt of the forestry image is. To note that, the images are taken with a fixed coordinate system, so it is not necessary to compute all the angle values because the author knows a priori that the image is between that angle values.

The results obtained by the authors for the second (final) approach described above were good, with a median angle error of 1 degree for the images tested. Also, the authors tested the algorithm over some images of urban scenes and the results were good as well. Although this algorithm seems simple and easy to implement, it is only applied to controlled images because of the optical nature of the problem.

In conclusion, this paper shows a simple image processing approach that, with some adjustments can be used to detect the tilt angle in the background in FotoFaces system.

Lifeng Pan et al. [41] propose a new image correction algorithm using an improved version of Hough Transform.

Basically, the authors first explained all the basic concept´s that are used in the tilt correction algorithm proposed, in this case, converting an image to gray scale, transforming the image into a binary one, i.e, image binarization, apply geometric space transformations, applying the Hough transform and finally computing the tilt angle and rotation necessary to correct the image.

The improvement in this Hough transform algorithm is in the fact that the tilt angle of the image is chosen using only the maximum value of all the maximum local values of each straight line of the original image in the parameter space, i.e., using the polar coordinate angle and radius that correspond to this maximum value. Then the counter rotation is made using a double linear interpolation gray-level method to restore the loss of the pixels of the rotated image.

Although this method seems simple and the results shown in the paper are good, the authors only show 2 images with simple objects, which is not a guarantee that this algorithm can work well to detect complex background tilt angles that will be evaluated in FotoFaces system.

Jun-Tae-Lee et al. [42] propose a “new” semantic line detection algorithm using CNN with multi-tasking learning using both classification and regression, named as SLNet.

The authors use the convolutional and max pooling layers to get the image feature maps for the input images. To note that the 13 convolutional layers are from the VGG16 network. After, they developed a line pooling layer to extract the line candidates feature vectors of the input image and finally, they used the feature vectors to classify and in case of a semantic line to estimate the exact position of each line candidate, in this case using the regression and classification layers respectively. To note that, when this work was developed/published (2017) there were almost no information or related works about semantic line detection, so the authors do not have many comparisons works that can be used.

The dataset used to train, validate, and test the SLNet was also developed by the authors and is named as Semantic Line Dataset (SEL). This dataset provides the input image for the network as well as the ground-truth semantic lines for each image that were chosen by 6 subjects that have basic knowledge in photography composition. Also, it is composed by 1750 outdoor images that were taken from some photo-sharing websites where 61% of the images have got multiple semantic lines (the rest of the photos only have one semantic line).

In this proposed SLNet model there are 2 modes for semantic line detection: the primary semantic line detector and the multiple semantic line detector. As the name say, the primary semantic line detector detects only the most "important"/relevant semantic line of the image while the second one detects all the semantic lines of the image.

To test the new model, the authors first do a qualitatively assessment where they check each image to see if the primary semantic line goes with some photographic composition rules such as horizon, diagonal, and symmetry. Also, they see if the multiple semantic lines represent efficiently the special layouts of each image.

Quantitatively, for the primary line detector, if the semantic line is confirmed by the softmax layer then the authors used the accuracy metric defined as the number of test images where the semantic line is correctly labeled (the Mean Intersection Over Unit (mIoU) score of the ground-truth and the predicted line is greater than a threshold value) divided by the total number of test images. In terms of the multiple semantic lines, the authors used a precision and recall scores, where the precision is the number of the correctly labeled semantic lines divided by the false positives plus the correctly labeled ones, and the recall is the number of correctly labeled semantic lines divided by the same number plus the false negative ones.

With all that, the authors varied some hyperparameters and layers of the network model, such as the pooling layers for example, and obtained for the best model an accuracy of 92.0% for the primary semantic line detection and 80.44% and 83.50% for the precision and recall of the multiple semantic line detector.

Finally, the authors show that this new model can be applied into 3 different areas of CV that are the horizon estimation, composition enhancement and image simplification. As the horizon estimation is the only one that have some sort of interest to implement in Fotofaces, it is only referred the results for that one.

In this case, the authors trained and tested the model over the Horizon Lines in the Wild (HLW) dataset that consists of around 14k images, where 2k are used for testing and the rest for training and validation of the network. The results obtained were only for primary line detection, where using the SLNet pre trained model, they obtained 70.5% of accuracy and using the same dataset to train the model the authors obtained 82.33% accuracy, which is better than the other 2 horizon detector methods mention in the paper with only 58.24% and 71.16% of accuracy.

This SLNet model can be useful in the horizon detection task for the FotoFaces system. Nevertheless, as it will be described further on, recent semantic line works (some are improvements from this one with the same authors) have better results.

Jun-Tae-Lee et al. [43] propose a novel semantic line detector using "mirror attention and comparative ranking and matching". The authors developed 3 different networks: one for mirror attention, named D-NET, and the other two for comparative ranking and matching, named R-Net and M-Net respectively.

In short, D-Net is used to extract the semantic lines, just like the SLNet in the previous paper [42], using regression and classification. Then, R-Net selects the most appropriate semantic lines by comparing then into pairs. Finally, the M-Net removes all the redundant lines that overlaps the most appropriate one.

In terms of Datasets used, the authors created another dataset named SEL_Hard, which contains more challenging images to detect semantic lines.

For evaluation purposes the authors used the same metric as in the previous work and outperformed the previous model SLNet in every metric.

Finally, they have made a study for two different task: dominant parallel lines and reflection symmetry axes.

For Dominant parallel lines, this new model was outperformed by two existing works when the angle tolerance was small, but for higher tolerances this model performed better.

In terms of Reflection symmetry axes, this new model outperformed all the literature works cited in the paper.

Jun-Tae-Lee et al. [44] proposed a novel algorithm for semantic line detection. This algorithm is composed by 2 networks, namely a Selection Network (S-Net) and a Harmonization Network (H-Net). In this case, the objective of this semantic line detector is to find the harmonious semantic lines, i.e, "a group of semantic lines in an image can be regarded as optimal, when they convey the composition of the image harmoniously".

In short, the S-Net model is used to compute the probabilities and offsets of the semantic lines. After that, a process to remove the irrelevant lines using the outputs of S-Net is made. Next, using the H-Net edge weights a complete graph is constructed. Finally, a maximal weight clique that represents the optimal semantic lines of the input image is determined.

Also, the authors introduce a new metric, named Harmony-based Intersection-over-union (HIoU), that allows to access the overall harmony of the detected lines.

With all of that, the authors reduce the computational complexity when comparing to his previous work algorithm DMR by a factor of 1/20 and achieve competitive results as well.

Finally, the authors used 4 different datasets: SEL, SEL_Hard, SLK5 and CULane to evaluate and compare with previous works and the new algorithm outperformed them in every single dataset. Also, they have made a comparison to other works that used SLK5 and CULane datasets and obtained better performance metrics too.

7.2 Background Dataset

As in the previous chapters, to test and evaluate the implemented algorithms a dataset was necessary. In this case, as the photos need to contain a person in it and be sort of profile pictures, the dataset was made using 50 images from the previous datasets. Also, to annotate this dataset was used 3 persons.

Two of the annotations were made considering that the images are considered tilted if the background plan is not parallel to the camera position. Otherwise, in case that the camera and the background plane were parallel then they analysed the lines of the background to say if the background was or was not tilted.

On the other hand, the other annotation was based on the image edges without the face region and all the region below the mouth line, that will be shown in the next section in Figure 7.2.

As it will be seen, this different type of annotation analysis will lead to different accuracy results.

In Figure 7.1 it is possible to observe some examples of the test Background Dataset. To note that this images are the input images and do not have the face rotation yet.

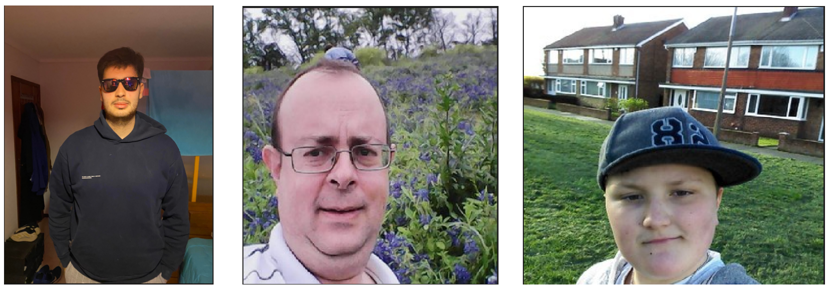


Figure 7.1: Examples of the Background Dataset

7.3 Foresty Tilt Correction Algorithm

In this section one of the solutions from the Literature review ([40]) is presented, namely the Foresty tilt angle correction.

This algorithm applies a rotation to the image. Then using the edges from that image (in the article they used Sobel Operator to compute the edges), the number of white pixels that form a vertical line using some threshold values for the maximum space between pixels is computed. Also, there is a threshold value for the minimum number of pixels to form a line. After applying all the rotations and computing every sum of white pixels that forms a line for each rotated image, the angle of the image is the one of the highest white pixels counted.

Although, this is a simple algorithm, some previous steps were incorporated (preprocessing steps) and some changes were made too to make the algorithm compatible with FotoFaces requirements.

With all of that, the final algorithm structure was the following:

- resize the candidate image to a scale less than 1000 pixels in both height and width
- detect the candidate face and rotate it (in case that the face is tilted) and obtain his face landmarks with the `Dlib` detector [45]
- use the HED [46] to obtain the edges from the image
- exclude the face region and all the region below the mouth line of the detected face from the edge image

- apply the Foresty tilt angle correction algorithm to obtain the angle of the background over the edge image with only the background

In Figure 7.2 it is possible to observe the image preprocessing steps before applying the Foresty tilt angle algorithm.

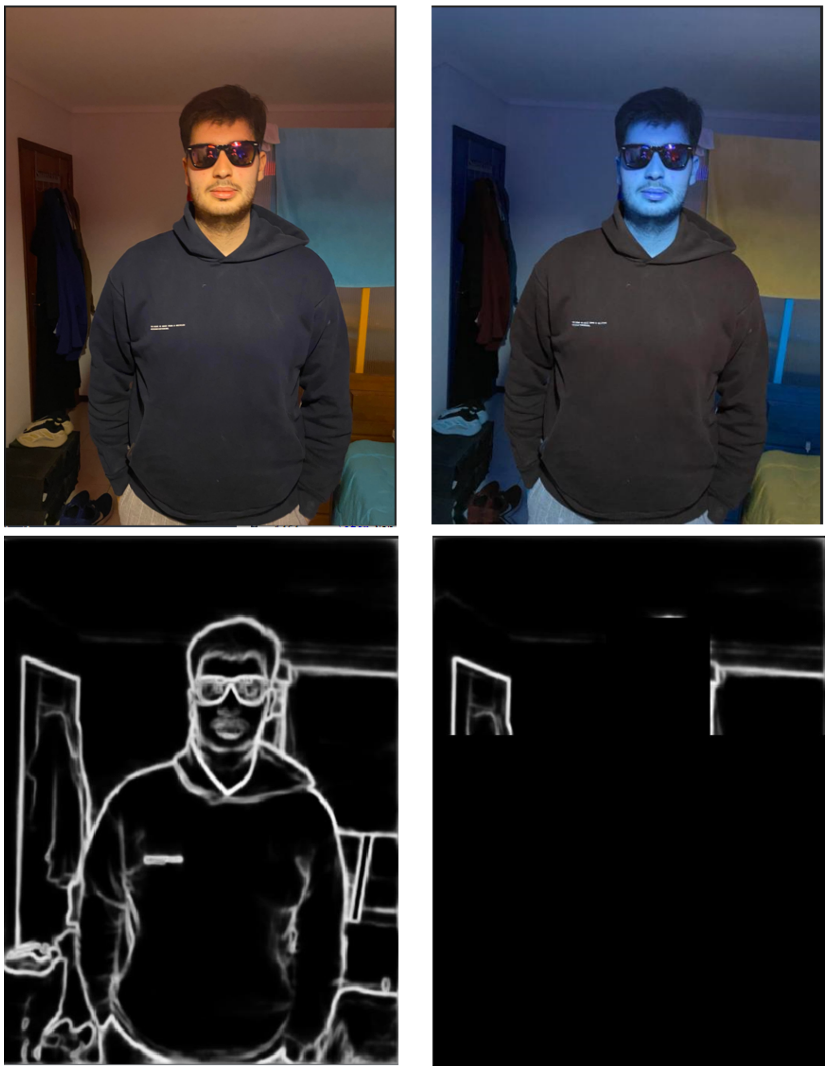


Figure 7.2: Preprocessing images steps of the Foresty tilt algorithm: Top-left - original image, Top-right - Rotated Image (RGB channels switched to acbgr), Bottom-Left - HED, Bottom-Right - HED without face region and all the region below mouth line

To determine if the image background was tilted or not tilted the following criteria was used:

- Not Tilted: -5 <= angle <= 5 or 85 <= angle <= 90 or -90 <= angle <= -85
- Tilted: rest of angles

The results obtained for 3 different annotations were the following:

- Annotation 1: 91% of accuracy
- Annotation 2: 60% of accuracy

- Annotation 3: 65% of accuracy

With this accuracy values it was possible to conclude that the annotation analysis have a lot of impact. In this case, as the annotations 2 and 3 were made following similar reasoning, then the final results were also similar, but they were really different from the accuracy result of annotation 1.

Another aspect to take into consideration over this algorithm was the running time to compute the image angle that was around 3 min for each image. Nowadays, waiting 3 minutes to update a profile photo is not a considerable option.

Although this algorithm has flaws and it is time consuming, it was chosen to be implemented in this version of FotoFaces system.

7.4 Semantic Line Detector

In [42] a semantic line detector network was shown, named SLNet. Although there were 2 more works, [43] and [44], related to the evolution of this network, in this section only the results of our test background dataset for the SLNet are shown due to reproducibility issues.

It was used the source code from the paper and the pre-trained paper network parameters, that is available in the paper github repository. Also, it was necessary to make some basic changes, like changing the directories (test dataset, parameters, etc...), outdated builtin functions from the libraries used (pytorch for example), among others.

Some of the results obtained are in Figure 7.3. As it can be seen in the top row, the images got understandable semantic lines, i.e., the detected semantic lines match the image, while in the bottom row, the detected semantic lines do not make sense at all. One of the reasons that can justify this fact is that the dataset used to train this network [42] is composed by outdoor background images, where there were none or small objects (in our case human beings), while our test dataset got in a first plane a person that occupies most of the image region and in some of the images the background is indoor. Also, it is given as input to the network the candidate(s) line(s), and to test the background images the first candidate line from the downloaded test file of the authors was used, i.e., this candidate line does not correspond to the ground-truth of the image given as input.

Making a global appreciation of this algorithm, it can be said that it is not a good option to be implemented in the FotoFaces system.

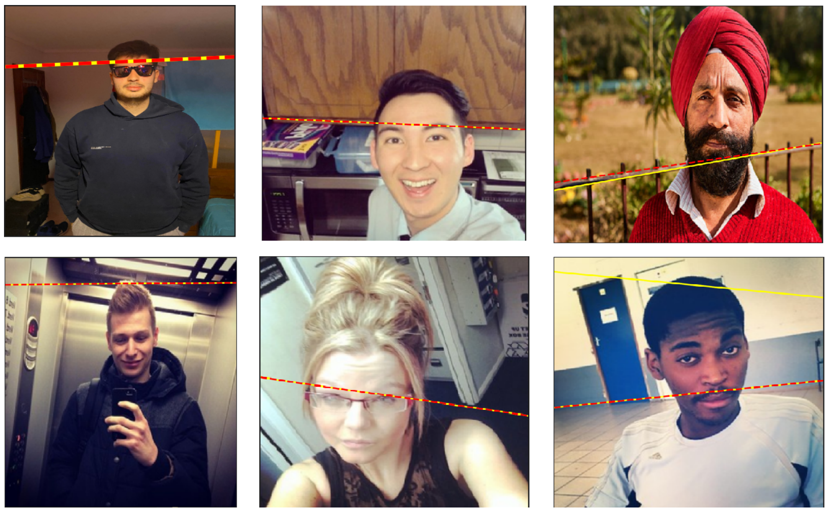


Figure 7.3: Examples of the output from the Semantic line detector SLNet (red dashed lines - primary lines , yellow lines - multiple lines)

7.5 Conclusions for Background Analysis

Both of the Background algorithms presented in this chapter were not a good option for FotoFaces system. It was concluded that it should be a better option to think about the parallelism between the camera and the background plane, instead of the just computing the angle and analyse the semantic/main line(s)/edge(s) of the image.

CHAPTER 8

Conclusions

During this thesis a lot of recent literature works were explored that could improve the FotoFaces system. Also, some of these methods were tested and in the case that the results achieved were good, they were implemented into the system software.

The 3 main contributions of this work were for Sunglasses Detection, Hat Detection and Background Analysis.

The Sunglasses and Hat detection followed data centric approaches, where 2 new datasets were created: Sunglasses dataset and Hat dataset.

The model used for Sunglasses and Hats detection was the VGG16 (tranfer learning). To note that, for the Sunglasses detection, some ablations studies were made in order to get the VGG16 model that was reused for the Hats detection.

To improve the performance of the model over each dataset, preprocessing methods were added to the dataset images before each stage: training, validation and testing. In this case the cropping over the intended regions was the most impacting preprocessing method.

The results obtained for sunglasses detection were good, with 98% of accuracy using the final test dataset. For hats detection, the best accuracy values achieved was only of 89% over the test dataset (Hat dataset) and 96% for the final test dataset that only contains 4 images with hats.

With that, it is possible to conclude that the sunglasses model and the respective dataset are robust enough and the model can predict well if a person is wearing sunglasses or not in an image. Nevertheless, the Hat Detection model and dataset can still be improved (mainly the dataset).

In terms of Background Analysis, 2 literature works were implemented and tested, with the appropriate adjustments.

The objective of these 2 works was to analyse the lines in an image and compute the tilt angle. With that, a small background test dataset was made using some images from the previous datasets. The first work was a low-level approach that was used primarily to compute the tilt angle of a Forest. The second work was a ML one, used to compute the semantic lines of an image.

The results obtained for both of these works were not satisfying at all and led to the conclusion that the problem statement was not correctly understood. The focus of the background analysis should have been on the parallelism of the background and the camera and not over the lines/edges of the image.

8.1 Future Work

Taking into account the current version of FotoFaces system, some of the explored topics could be improved, like:

- Face Detection, Alignment and Recognition - use recent and robust models that have been shown to outperformed the current ones
- HPE - use recent and robust models that have been shown to outperform the current one
- Hat detection - make an approach more Model Centric to improve the accuracy values and/or increase the current dataset (Data Centric)
- Background Analysis - study/implement models that first check the parallelism between the camera and the background plane

References

[1] *Dlib documentation*, http://dlib.net, Accessed: 2021-12-23.

[2] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe, "Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment," *IEEE Transactions on Image Processing*, vol. 29, pp. 4041–4056, 2020.

[3] ——, "Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment," *IEEE Transactions on Image Processing*, vol. 29, pp. 4041–4056, 2020.

[4] W. Zhang, K. Ma, G. Zhai, and X. Yang, "Learning to blindly assess image quality in the laboratory and wild," in *IEEE International Conference on Image Processing*, 2020, pp. 111–115.

[5] Chamandeep Vimal and Neeraj Shrivastava, "Face detection's various techniques and approaches: A review," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 10, Jan. 2022. DOI: 10.22214/ijraset.2022.39890.

[6] Rein-Lein Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, May 2002.

[7] Huaizu Jiang and Erik Learned-Miller, "Face detection with the Faster R-CNN," *IEEE 12th International Conference on Automatic Face & Gesture Recognition*, 2017. DOI: 10.1109/FG.2017.82.

[8] Meihua Gu, Xiaolong Liu, and Jing Feng, "Classroom face detection algorithm based on improved MTCNN," Jan. 2016.

[9] Yanda Meng, Xu Chen, Dongxu Gao, Yitian Zhao, Xiaoyun Yang, Yihong Qiao, Xiaowei Huang and Yalin Zheng, *3D Dense Face Alignment with Fused Features by Aggregating CNNs and GCNs*, https://arxiv.org/abs/2203.04643, Accessed: 2022-04-20, Mar. 2022.

[10] Sharma S, Karthikeyan Shanmugasundaram, and Sathees Kumar Ramasamy, "FAREC - CNN Based Efficient Face Recognition Technique using Dlib," *IEEE 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2016.

[11] Andrea F. Abate, Carmen Bisogni, Aniello Castiglione, Michele Nappi, "Head pose estimation: An extensive survey on recent techniques and applications," *Elsevier: Pattern Recognition*, Feb. 2022. DOI: 10.1016/j.patcog.2022.108591.

[12] Xiao Li, Dong Zhang, and Ming Li, "Accurate head pose estimation using image rectification and a lightweight convolutional neural network," *IEEE Transactions on Multimedia*, 2021. DOI: 10.1109/TMM.2022.3144893.

[13] Naina Dhingraa, *Lwposr: Lightweight efficient grained head pose estimation*, https://openaccess.thecvf.com/content/WACV2022/html/Dhingra_LwPosr_Lightweight_Efficient_Fine_Grained_Head_Pose_Estimation_WACV_2022_paper.html, Accessed: 2022-04-18, 2022.

[14] K. Shorten C., "A survey on image data augmentation for deep learning," *Journal of Big Data*, pp. 6–60, 2019. DOI: 10.1186/s40537-019-0197-0.

[15] Q. V. Barret Z, "Neural architecture search with reinforcement learning," Feb. 2017. DOI: 10.48550/arXiv.1611.01578.

[16] *The impact of pre-processing algorithms in facial expression recognition*, https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11605/116051Q/The-impact-of-pre-

processing-algorithms-in-facial-expression-recognition/10.1117/12.2587865.short?SSO=1, Accessed: 2022-04-02.

[17] *Glass detetion project with the same dataset*, https://www.kaggle.com/code/jorgebuenoperez/computer-vision-application-of-cnn, Accessed: 2022-03-28.

[18] *A robust deep learning approach for glasses detection in non-standard facial images*, https://www.researchgate.net/publication/347260372_A_robust_deep_learning_approach_for_glasses_detection_in_non-standard_facial_images, Accessed: 2022-02-10.

[19] A. M. Basbrain, I. Al-Taie, N. Azeez, J. Q. Gan, and A. Clark, "Shallow convolutional neural network for eyeglasses detection in facial images," in *2017 9th Computer Science and Electronic Engineering (CEEC)*, 2017, pp. 157–161. DOI: 10.1109/CEEC.2017.8101617.

[20] S. Wang, Z. Liu, S. Lv, *et al.*, "A natural visible and infrared facial expression database for expression recognition and emotion inference," *Multimedia, IEEE Transactions on*, vol. 12, pp. 682–691, Dec. 2010. DOI: 10.1109/TMM.2010.2060716.

[21] B. Wu, H. Ai, and R. Liu, "Glasses detection by boosting simple wavelet features," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 1, 2004, 292–295 Vol.1. DOI: 10.1109/ICPR.2004.1334110.

[22] *Multidimensional classification using luts*, https://github.com/ubuntuslave/ML-Classification_from_LUTs, Accessed: 2022-01-29.

[23] *Wavelet scattering from mathworks*, https://www.mathworks.com/help/wavelet/ug/wavelet-scattering.html, Accessed: 2022-01-29.

[24] E. Gumus, N. Kilic, A. Sertbaş, and O. Ucan, "Evaluation of face recognition techniques using pca, wavelets and svm," *Expert Syst. Appl.*, vol. 37, pp. 6404–6408, Sep. 2010. DOI: 10.1016/j.eswa.2010.02.079.

[25] P. J. Phillips, H. Moon, S. Rizvi, and P. Rauss, "The feret evaluation methodology for face-recognition algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1090–1104, Oct. 2000. DOI: 10.1109/34.879790.

[26] S. Asteriadis, N. Nikolaidis, A. Hajdu, I. Pitas, "An eye detection algorithm using pixel to edge information," May 2014.

[27] K. Messer, J. Matas, J. Kittler, J. Luettin and G. Maitre, "The extended m2vts database.," *Second International Conference on Audio and Video-based Biometric Person Authentication*, vol. 22, pp. 72–77, Mar. 1999.

[28] *Bioid database*, http://www.bioid.com/downloads/facedb/facedatabase.html, Accessed: 2022-03-24.

[29] *Glasses or no glasses dataset from kaggle*, https://www.kaggle.com/jeffheaton/glasses-or-no-glasses?fbclid=IwAR1VZY88NadddywT2Mk2NxRqLnDL_iAvy1tV87mJjK2O-YwHZldK1xP8iMM, Accessed: 2021-01-17.

[30] *Glasses Detection - OpenCV & DLIB*, https://medium.com/mlearning-ai/glasses-detection-opencv-dlib-bf4cd50856da, Accessed: 2021-01-17.

[31] *Sunglasses /no sunglasses database*, https://www.kaggle.com/datasets/amol07/sunglasses-no-sunglasses, Accessed: 2022-03-25.

[32] *Step by step vgg16 implementation in keras for beginners*, https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c, Accessed: 2022-04-02.

[33] *Image quality assessment : Brisque*, https://learnopencv.com/image-quality-assessment-brisque/, Accessed: 2021-12-31.

[34] Rahul N. Bhadeshiya, Dr. K. N. Brahmbhatt and Dr. J. R. Pitroda, *Hard-hat Detection using YOLOv4*, https://ieeexplore.ieee.org/document/9532896, Accessed: 2022-04-19, 2021.

[35] Nikolav Filatov, Natalia Maltseva Aleksandr Bakhshiev, *Development of Hard Hat Wearing Monitoring System Using Deep Neural Networks with High Inference Speed*, `https://ieeexplore.ieee.org/document/9208155`, Accessed: 2022-04-19, 2020.

[36] *Hijaab and hats dataset from kaggle*, `https://www.kaggle.com/datasets/asif85/hijaab-and-hats-dataset`, Accessed: 2021-06-21.

[37] *Selfie-image-detection-dataset from kaggle*, `https://www.kaggle.com/datasets/jigrubhatt/selfieimagedetectiondataset`, Accessed: 2021-06-21.

[38] *Selfie data set from ucf (center of research in computer vision)*, `https://www.crcv.ucf.edu/data/Selfie/`, Accessed: 2021-06-21.

[39] *Women with caps from unsplash*, `https://unsplash.com/s/photos/women-with-caps`, Accessed: 2021-06-21.

[40] Jaroslav Matej, "Determination of forestry machine's tilt angle using camera and image processing," *ELSEVIER Computers and Eletronics in Agriculture*, vol. 109, pp. 134–140, 2014. DOI: `10.1016/j.compag.2014.09.011`.

[41] Lifeng Pan and Fu Liu and Dangui Chen, "Image correction scheme based on improved hough transform," *Rev. Téc. Ing. Univ. Zulia*, vol. 39, no. 10, pp. 392–400, 2016. DOI: `10.21311/001.39.10.46`.

[42] Jun-Tae Lee, Han-Ul Kim, Chul Lee, and Chang-Su Kim, "Semantic line detection and its applications," in *ICCV*, 2017.

[43] J.-T. L. Dongkwon Jin and C.-S. Kim, "Semantic line detection using mirror attention and comparative ranking and matching," in *ECCV*, 2020.

[44] D. Jin, W. Park, S.-G. Jeong, and C.-S. Kim, "Harmonious semantic line detection via maximal weight clique selection," in *CVPR*, 2021.

[45] *Dlib face detector python example*, `http://dlib.net/face_detector.py.html`, Accessed: 2021-12-27.

[46] *Holistically-nested edge detection with opencv and deep learning*, `https://www.geeksforgeeks.org/holistically-nested-edge-detection-with-opencv-and-deep-learning/`, Accessed: 2022-06-20.

[47] *Rbg and hsv color models*, `https://www.researchgate.net/figure/a-the-RGB-color-space-black-arrows-show-the-three-main-color-dimensions-whose-values_fig2_323952018`, Accessed: 2021-12-23.

[48] *Grayscale color model*, `https://www.researchgate.net/figure/a-the-RGB-color-space-black-arrows-show-the-three-main-color-dimensions-whose-values_fig2_323952018`, Accessed: 2021-12-23.

[49] *Hsv color map*, `https://coderedirect.com/questions/268559/how-can-i-only-keep-text-with-specific-color-from-image-via-opencv-and-python`, Accessed: 2021-12-23.

[50] *Histogram oriented gradients opencv example*, `https://learnopencv.com/histogram-of-oriented-gradients/`, Accessed: 2021-12-23.

[51] *Histogram oriented gradients*, `https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients`, Accessed: 2021-12-23.

[52] *Linear classifier*, `https://en.wikipedia.org/wiki/Linear_classifier`, Accessed: 2021-12-26.

[53] *Image pyramids*, `https://docs.opencv.org/4.x/dc/dff/tutorial_py_pyramids.html`, Accessed: 2021-12-27.

[54] *Sliding windows for object detection with python and opencv*, `https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/`, Accessed: 2021-12-27.

[55] *Facial landmarks with dlib, opencv and python*, `https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/`, Accessed: 2021-12-26.

[56] *Euler angles*, `https://en.wikipedia.org/wiki/Euler_angles`, Accessed: 2021-12-27.

[57] *Head pose estimation using opencv and dlib*, https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/, Accessed: 2021-12-27.

[58] *Camera calibration mathworks*, https://www.mathworks.com/help/vision/ug/camera-calibration.html, Accessed: 2021-12-27.

[59] *Camera calibration opencv tutorial*, https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, Accessed: 2021-12-27.

[60] *Pinhole camera model*, https://en.wikipedia.org/wiki/Pinhole_camera_model, Accessed: 2021-12-28.

[61] *Pinhole camera model*, https://hedivision.github.io/Pinhole.html, Accessed: 2021-12-28.

[62] *Opencv rodrigues function*, https://shimat.github.io/opencvsharp_docs/html/0091381a-4571-5241-4c69-ff8d0678e5b5.htm, Accessed: 2021-12-29.

[63] *Opencv decompose porjection matrix function*, https://shimat.github.io/opencvsharp_docs/html/3513d2db-c217-11f4-5419-e75d7f87e799.htm, Accessed: 2021-12-29.

[64] *Eye blink detection with opencv, python, and dlib*, https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/, Accessed: 2021-12-30.

[65] *Image kernels*, https://setosa.io/ev/image-kernels/, Accessed: 2021-12-30.

[66] *Kernels (image processing)*, https://en.wikipedia.org/wiki/Kernel_(image_processing), Accessed: 2021-12-30.

Appendices

In order to progress and improve the FotoFaces system it is necessary to have a look of the initial state of work and understand the basics behind it.

In this Appendices it will described all the concepts that were used/applied in the first software developed for FotoFaces by Daniel Canedo. Also, the FotoFaces Application Programming Interface (API) is described in one of the sections. All the support articles that were developed within the scope of other academic subjects that were useful to develop the rest of the FotoFaces system are present into the github repository that is shown in the last section.

Colour Photo

As it was said in the Introductory chapter, FotoFaces follows a series of criteria in order to work and approve a submitted photo by the user.

The first criteria is very simple to test.

Before jumping into the solution implemented into the software, let´s have a look into the concept of ColorSpace and some different types of color spaces that can be used in an image.

Color Space and Color Model

Basically a Color Space is the way that a set of colors are organised. It can be represented by a Color Model, that is the mathematical model of a specific Color Space. This mean that colors can be represented by numerical values.

RGB

For example RGB color model has got three channels, which are relative to the Red, Green and Blue colors. Each one of the channels has got 8 bits, being in the total a model with 24 bits. When combined they can generated a lot of different colors. A simple way of showing this color model is through the Figure 1, where it is possible to see a cube with the 3 different components being varied on the left side of the Image. It is important to note that the range of the numerical values for the 3 components goes from 0 to 255 (256 values) and the maximum values of the 3 components (R = 255, G = 255, B = 255) represent the white color, while the minimum values (R = 0, G = 0, B = 0) represent the Black color.

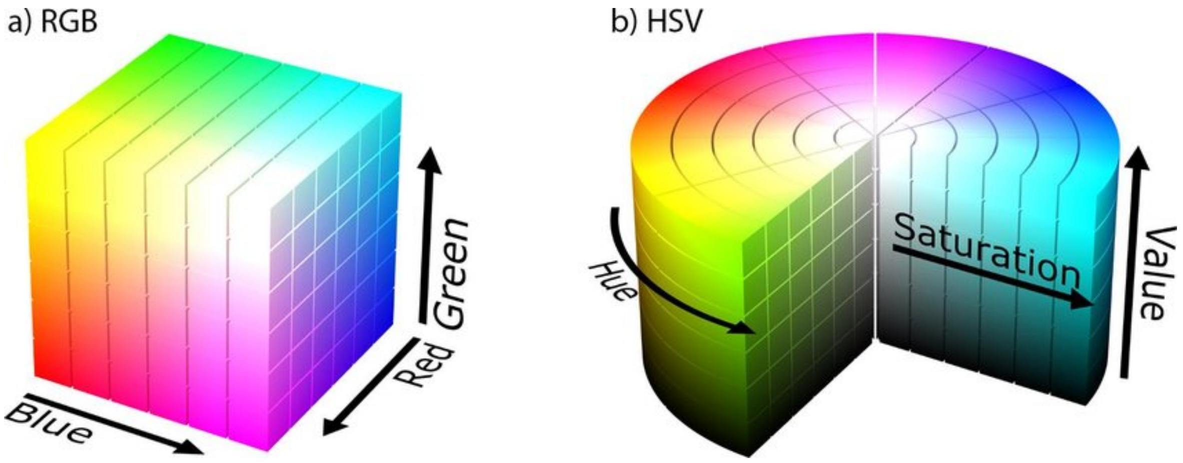


Figure 1: RGB and HSV color models [47]

Grayscale

Other color model that is used in many applications is the Grayscale model. This one only got one channel (8 bits), which represents the amount of light in an image (each pixel of the image has information about the intensity of light) and it´s range goes from 0 to 255 (256 values). For lower values of intensity the image gets darker and for higher values the image get brighter. The 0 value is the black color and the 255 value is the white color. The Figure 2 ilustrates the Grayscale color model.

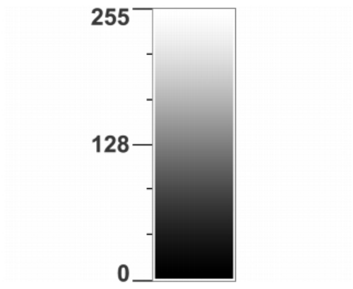


Figure 2: Grayscale Color Model [48]

HSV

Another color Model that is commonly used is the HSV model. This one has got 3 channels, as the RGB, but they have different meanings. The H is for Hue, S is for Saturation and V is for Value. Usually it is referred as Hue, Saturation, and Brightness (HSB) where H and S are the same that in HSV, and the B means Brightness. Basically the Hue verifies the color type, usually going from 0 to 360 degrees as it can be seen in Figure 1. Also it can be represented from 0 to 180. That is shown in HSV color map in Figure 3. The saturation value, also know as purity, determines how pure is the color. This means that, the lower the value for saturation, the more grayscale the image will be. In the color map of Figure 3 is the y axis that represents the Saturation value. The range goes from 0 to 255. Finally the V (Value) of HSV is the bright of the image. In color map goes from 0 to 255 value, however it is not possible to see it because it has only 2 axis (2 dimensional map). In the color model Figure 1 it´s shown the effect of V in HSV model.

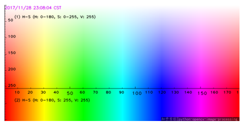


Figure 3: HSV Color Map [49]

The solution that was used into the software to verify is the image submitted of the candidate has color was simply separate and comparing the 3 RGB channels of the image. Basically if the 3 channels were equal, this means that the image was in grayscale. If they were not equal the image has colors. So the first criteria was completed with success.

Face-detection

`Dlib`

In order to verify if the image has a face in it, it was used a C++ library with an extension for Python named `Dlib` [1]. `Dlib` is a library with ML algorithms and tools that are useful when using them in real world applications. It was used some functions of this library like `get_frontal_face_detetor` and `shape_predictor`. The `get_frontal_face_detetor`, as the name said, is used to detect a frontal face in a given input image. To note that the image is given in grayscale instead of RGB. To detect the face this function used a HOG combined with a linear classifier, a Image Pyramid and a sliding window detection scheme.

HOG

An HOG [50], [51] is a feature descriptor, i.e, is a representation of an image using only the useful information in it, used in CV and image processing with the purpose of detecting objects.

Briefly this technique counts the number of occurrences of the oriented gradient in specific localized zones in a image. It is a similar technique when comparing to edge orientation histograms, scale-invariant feature transform descriptors and shape contexts, but it is calculated from a dense grid of uniformly spaced cells and uses a superimposed local contrast normalization in order to achieve better accuracy.

Let´s have a look into a more detailed explanation. Basically, the features used by the HOG are the histograms of oriented gradients. This are really important, because when applying the gradient in an image, x and y derivatives, the magnitude of that gradient can tell if that zone of the image is a edge/corner or a flat region. For a edge/corner the magnitude of the gradient is higher, because a gradient is a vector that indicates the direction and sense in which the higher rate of change occurs. In opposition, the flat regions have a low magnitude gradient. With this magnitude information, it is possible to use only the edges/corners, that gives a lot of information about an image when comparing to flat regions.

Usually to calculate an HOG, it is applied image preprocessing first, to ensure that that color and gamma values are normalized. Although, its impact is not very significant when comparing to an image without preprocessing, so it will be not cover here.

With that, the “first step” to calculate an HOG is to compute the gradients of an image. In order to do that, is it necessary to calculate the vertical, x, and horizontal, y, gradients through some filtering kernels, like the ones in Figure 4.

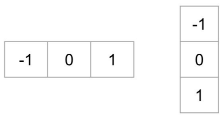


Figure 4: Filtering Kernels [50]

After that, it is calculated the magnitude and direction of all gradients using the respective formulas shown below:

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan\left(\frac{g_y}{g_x}\right)$$

Finally, since every pixel have the gradient magnitude and direction it is now possible to compute the HOG. To note that, as a RGB color image has 3 channels, the magnitude of the gradient at each pixel is the higher magnitude value of the 3 channels and the respective direction value (angle), is the angle of that maximum magnitude value.

In order to compute the HOG, the image is divided into cells where each cell contain the same amount of pixels and for each cell is computed a histogram of gradients.

This histogram of gradients is calculated using the weighted vote of each pixel, where the bins of the histogram goes from 0 to 180 degrees or 0 to 360 degrees depending if the gradient is unsigned or signed respectively. For human detection it was seen that unsigned gradients are generally better.

Although this seems a good solution, the gradients suffer from lighting problems,because if it is used the same image but 2 times darker, then the gradients magnitude values will be 2 times smaller (half) and, therefore, the histogram values will be half of the previous one. A simple solution to resolve this problem is applying a block normalization over the histograms. This makes the HOG magnitude values being independent from the scale, i.e., it does not matter if the image is lighter or darker that the normalized magnitude values will be always the same. For example, lets consider a RGB color vector with [32,128, 64]. The L2 norm of this vector is given by the square root of each squared value which is equal to 146.64. Dividing each value by this norm it is obtained the normalized vector [0.22, 0.87, 0.43]. If it is used a RGB vector with 2 times the values from the previous one, [64, 256, 128] and it is computed the normalized vector again, it is possible to observe that the normalized vector of this 2 times RGB vector is the same as the previous normalized vector. This proves what was said above.

Last but not least, it is computed the final HOG of the image making a concatenation off all the normalized vectors. The Figure 5 illustrates a HOG of an athlete (human detection), where it can be seen that the direction of the HOG shapes the person, especially in the legs and torso of the athlete.

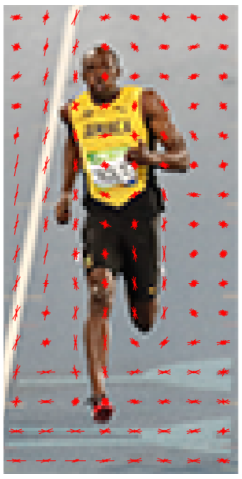


Figure 5: HOG over an image of an athlete [50]

Linear Classifier

In ML area, a classifier [52] serves, as the name suggest, to classify a object that belongs to a certain label (class/group) through some characteristics called features. So, a linear classifier, classifies an object through the linear combination of its characteristics. This classifiers works well for Document classification and for problems with a lot of features. This gives an advantage comparing to the non-linear classifiers that take a lot of time and computational power to classify an object with similar accuracy (precision) as the linear classifier.

Image Pyramids

An image pyramid, as the name tells, is a pyramid of the same image with different resolutions. Briefly the images are stored in a stack, with the higher resolution image in the bottom of the stack and the lower resolution image in the top of the stack. This is very useful when the objective is to find a object in a image, in this case a face, because it is not know at first the size of the object and then it is search the object in the different resolution images.

In Figure 6 it is possible to observe a pyramid composed by 4 levels (4 images).

Also, these pyramids are really useful too when using image blending, Figure 7, i.e., when it is made the join of two images.

There are two types of Image Pyramids that are: Gaussian Pyramids and Laplacian Pyramids.

Relatively to the Gaussian Pyramids, it´s higher level is formed by the consecutive removal of rows and columns of the higher resolution image, the one in the bottom. So, every time that it is ascend a level in the pyramid, each pixel is formed by the contribution of 5 pixels of the lower level being calculated by the Gaussian weight´s of that pixels. Basically, an

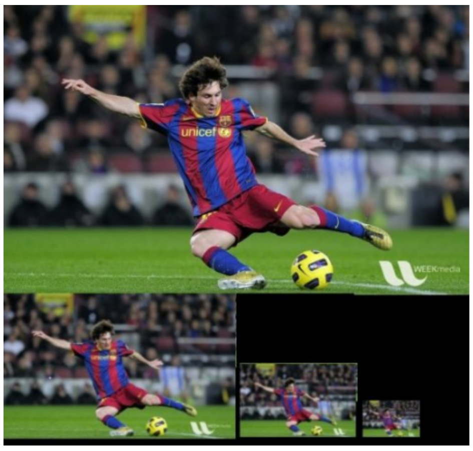


Figure 6: Image Pyramid with 4 levels [53]

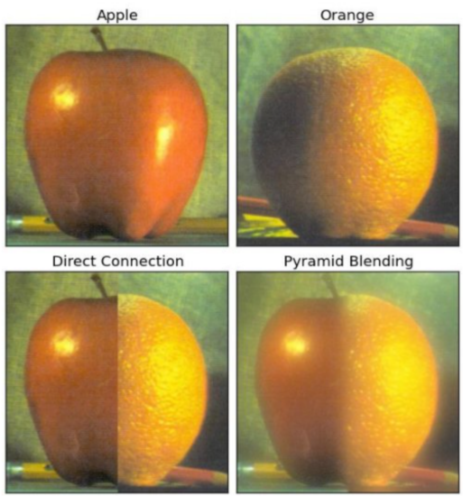


Figure 7: Image Blending using image pyramids [53]

image that has resolution M x N becomes an image with resolution M/2 X N/2, i.e, half of the previous image resolution with only a quarter of his area. The opposite occurs when it starts from the top of the pyramid, i.e, when it is going down in the levels of the pyramid the image becomes with four times more area than the top one. The Figure 6 is an example of a Gaussian Pyramid.

The Laplacian Pyramids are generated through the Gaussian Pyramids. This pyramids are edge images, i.e., it can only be seen the edges of the original image and most of the elements of the image are zeros (black). Normally this type of pyramids is used in image compression. A level of the Laplacian pyramids is formed through the difference between the image of that level of the Gaussian Pyramid and the image of the level above it. On Figure 8 it is possible to observe the Laplacian Pyramid using the previous Gaussian Pyramid of Figure 6.

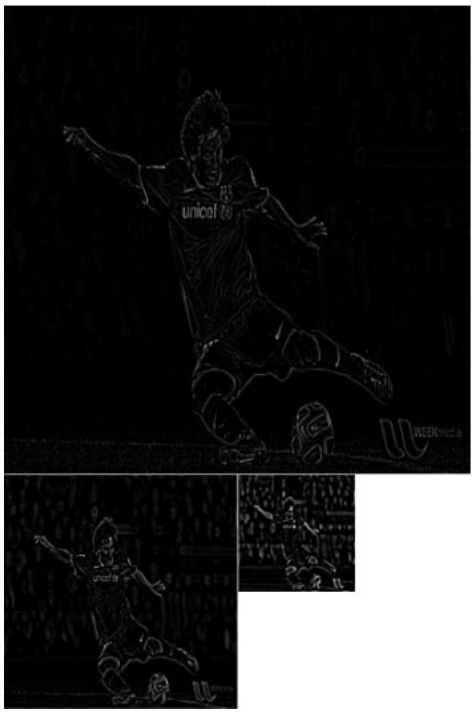


Figure 8: Laplacian Pyramid [53]

Sliding Window

In CV, a sliding window [54], as the name suggest, is a rectangular window with fixed height and width that travels across an image. So, when combined with a classifier, it is possible to detect if a desire object is inside the window, in this case, a face.

With all of that said, using the output of the `get_frontal_face_detetor function`, it is obtained the number of detected faces of the submitted image and the corresponding BB of each one. In this case, it is not verified yet if there are two or more detected faces, i.e, it is assumed to the candidate only submit images with one face and that´s corresponds to the largest BB detected, meaning that the smallest BB´s are false positives.

Now that it is known the BB of the face, i.e, it is stored the information with the Top, Left, Right and Bottom coordinates of the BB, it is made the landmarks prediction of that face using the `shape_predictor Dlib` function with the input file argument `shape_predictor_68_face_landmarks.dat` also from the `Dlib` library.

Facial Landmarks

Landmarks, more specifically facial landmarks, are points that localized regions of a face. Normally this points corresponds to the eye, nose, eyebrows, mouth and jaw locations. This landmarks are really useful for several CV applications like face alignment, HPE, face swapping, blink detection and others...

`Dlib` library has a function that allows to detect the facial landmarks of a face through estimation of 68 points of (x,y) coordinates. This facial landmarks can be seen on Figure 9

Finally the Function `detect_face` returns a numpy array with the coordinates of the detected face landmarks. In order to do that, it is made a cast from `Dlib` data type to numpy

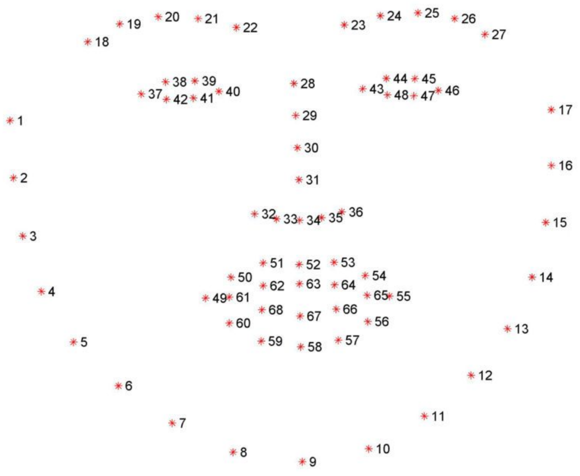


Figure 9: `Dlib` 68 Facial Landmarks Coordinates[55]

array. To note that through the software exists a lot of casts from a library data format into another library data format, but they will not be explained here since that have not significant importance.

Face Alignment

After detecting the Face and its landmarks, one important thing to do is Face Alignment, i.e, apply a rotation to the image if the Face is not straight.

In order to do that, is was develop a function named `rotate`, that applies a rotation to the candidate image using the previous computed landmarks. This rotation aligns the face with the axis of the image so that the eyes are perpendicular to the y´s axis and parallel to the x´s axis. So, the first step is to calculate the maximum distance between the eyes, i.e, using the left corner of the left eye, in Figure 9 is the coordinates of the 37 point, and the right corner of the right eye, the 46 coordinates in the same Figure, and the rotation angle between them, using the tangent arc between the y and x distance of the eyes. After, as the referential has the y axis "growing" down, then subtracts 180 degrees to the rotation angle calculated previously. With all of that done, it is computed the rotation matrix of the image using the rotation angle and then it is applied the rotation to the image. To note that the rotation is made in the clockwise if the rotation angle is negative and in counter clockwise if the rotation angle is positive. This function return as output the new align image and its landmarks.

One of the problems of this Face Alignment is that is not checked if the background and the Face are align too.

2D Transformations

Rotation Matrix

As the name said, a Rotation Matrix is a transformation matrix that allows to apply a rotation to an object in Euclidean space. In our case, the rotation matrix (2 dimensions) comes in the following form:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

In order to apply the rotation to the image using a 2D Rotation Matrix is it made a matrix multiplication between the coordinates of the object and the rotation matrix.

Cropping

Now that the Face alignment is applied, it is needed to crop the image to the “final” PACO image.

First, it is calculated the “middle” x and y coordinates of the face. Going Back to Figure 9, it is used for “middle” x coordinate the average of x coordinates of points 1 and 17 and for the “middle” y coordinate the average of y coordinates of points 20 and 34 which corresponds ,more and less, to the nose region below the eyes. Then, it is calculated the distance between point 1 and 17, i.e, the face width. Now, it is added and subtracted the distance to the middle x and y coordinates in order to obtain 4 new points that are the limits of the new cropped image. If for some reason the limits are larger than the previous image, the function return a None parameter, meaning that the Face is not centered in the image or it is too close to the camera.

Finally it is made a resize of the image to a 500 x 500 resolution.

Face Recognition

The Face Recognition is very important, because with that it is possible to identify if the candidate person is the same as the one in the old image, if it exist one. Sometimes it can be a difficult task to verify, because some photos can be very old compared to the new ones, for example a person that has submitted a photo when he has 18 years old and now is submitting a new photo with 60 years old. This is a aging problem, but it can occurs other type of problems. This is a very interesting topic that can be explored and add to the system functionality.

So, coming back to the already implemented Face Recognition function, it uses a `Dlib` trained model to compare the old and the new candidate photo. First it uses the `Dlib` function `get_face_chip` that uses the image and the landmarks of the new candidate photo to return the face in the image into a numpy array in a upright position and scaled 150 x 150 pixels. Then, using a face recognition model file, named `dlib_face_recognition_resnet_model_v1.dat`, and the respective function `face_recognition_model_v1 and compute_face_descriptor`, it is calculated the the 128D vector of the face descriptor of the candidate.

With that done, now it is computed the reference face landmarks (old photo landmarks), using the same method that was used for the candidate image talked above and computed the old 128D old face descriptor as well. The last step consists in calculating the Euclidean distance between the two 128D face descriptors and the Face recognition function returns this Euclidean distance. If the Euclidean distance is less than 0.6, this means that the new candidate photo has got the same person that the old photo.

Other important criteria that the candidate photo needs to have is a frontal face position. In order to verify the face position it is needed to estimate the head pose, i.e., estimate the Euler angles (Roll, Pitch and Yaw).

Euler Angles

Euler angles are the 3 angles used “to describe the orientation of a rigid body with respect to a fixed coordinate system” [56]. These angles can be observe on Figure 10.

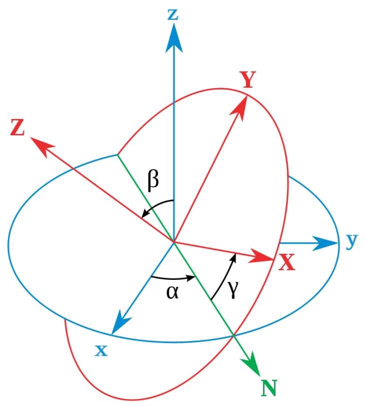


Figure 10: Euler Angles [56]

Basically, the 3 angles can generated two different types of rotations that are: extrinsic and intrinsic. Extrinsic rotations occurs when the 3 rotations are made to the reference coordinate system. On the other hand, intrinsic rotations are made in every new coordinate system, this means, every time a rotation occurs, the xyz coordinate system changes with the body movement and changes its orientation.

Now considering that these 2 methods of rotation are not used, there are 12 different possible rotations that are in 2 different groups:

- Proper Euler Angles (z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y).
- Tait-Bryan Angles (x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z).

The main difference between these two groups is that the first and the third rotation on the proper Euler angles is made using the same axis, while in Tait-Bryan Angles this rotations (first and third) are made into different axis. This means that, the line of nodes in the proper Euler Angle is the intersection of 2 homologous Cartesian Planes and this planes are parallel when the Euler angles are zero. For the Tait-Bryan angles, is the intersection of 2 non-homologous planes and they are perpendicular when the Euler Angles are zero.

The group that it will be focused is the second one, this is, the Tait-Bryan Angles ,also called Roll, Pitch and Yaw. To note that the different convention names given to the Tait-Bryan angle depends on the angle signs used.

The Roll, Pitch and Yaw are angles used normally in aircraft, that in this case, these angles will be calculated considering a face. The Figure 11 illustrates these angles over an aircraft.

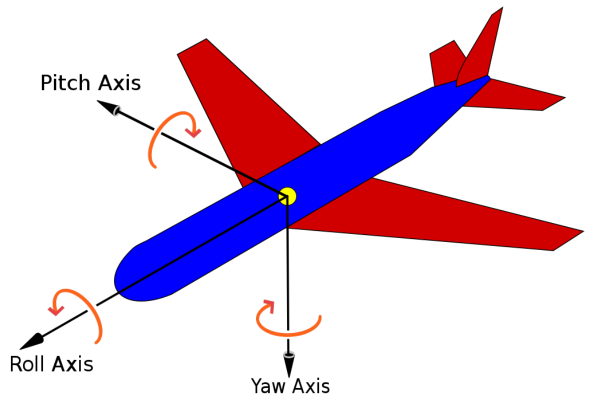


Figure 11: Euler Angles [56]

Pose estimation has been an important topic for CV because, as the name suggest, it allows to know how a certain object is oriented and what is its position relatively to a camera. In order to compute the pose of an object, in our case the face in the photo, it is needed to have the 2D projection points of the object and the corresponding 3D points of the same object in the world. Also, the pose of an object is composed by 6 numbers that are the rotation angles Roll, Pitch and Yaw talked above, and the translation numbers, X, Y and Z that represents the location of the object. These rotation and translations vectors are the camera extrinsic parameters.

So, first of all, it is needed to have the 2D and the 3D points of the face. In this case, the landmarks obtained before are the 2D points and the 3D points will be generic world coordinate points. To note that are only used a few points of the landmarks as well as the respective generic points to the 3D ones. This 3D generic world coordinate points [57] are the following:

- Nose: (0.0, 0.0, 0.0)
- Left eye: (-165.0, 170.0, -135.0)
- Right eye: (165.0, 170.0, -135.0)
- Left mouth corner: (-150.0, -150.0, -125.0)
- Right mouth corner: (150.0, -150.0, -125.0)
- Chin : (0.0, -330.0, -65.0)

Now that the 2D and 3D points are known (camera extrinsic parameters), it is needed to configure the intrinsic parameters of the camera.

Camera Calibration

As it was said before a camera can have extrinsic and intrinsic parameters [58] [59]. In this section it will be describe each one of them and also explained some 2 different types of distortion that can be generated by a camera. To note that the explanation is referred to a pinhole camera model [60] [61], that is a camera with no lens and a small hole, named pinhole. Is to that hole that the Light rays goes, in order to project an invert image of the world into the camera.

So, let´s understand how these intrinsic and extrinsic parameters are connected and can transform the "real world" into a 2D image. On Figures 12 and 13 it is illustrate a diagram that explains how the world coordinates are transform into the pixel coordinates and an image showing that. First, it is used the Extrinsic parameters, this is, rotation and translation vector in order to get the camera coordinates. After this, camera coordinates are transformed to the Pixel coordinates using the intrinsic Camera matrix, in this case named K, that is a matrix composed by the intrinsic parameters of the camera. These parameters are the focal length (fx,fy), that is the distance between the optical center, this is the camera in the 3D world (the pinhole) and principal point in the inverted image plane, that is the point in the same axis (principal axis) as the optical center, the principal point (cx, cy) and a skew coefficient (s), that is only different of zero if the image axis are not perpendicular. This intrinsic Camera matrix is shown on Figure 14 and the pixel skew on Figure 15 .

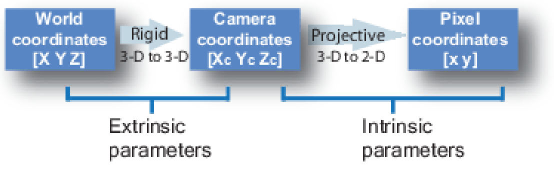


Figure 12: Diagram of Camera Calibration Parameters [58]

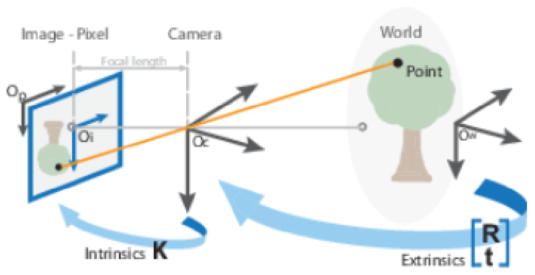


Figure 13: Example of Camera Calibration Parameters [58]

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

Figure 14: Intrinsic camera matrix [58]

Distortions are really common artifacts that occur because of the lens of the camera. It can result into two type of distortions that are the Radial and Tangential Distortions. To note that a ideal pinhole camera does not have this problem because it has got no lens, that´s why the previous camera matrix did not have this distortions into account.

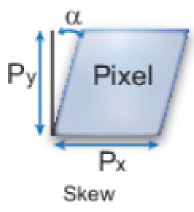


Figure 15: Pixel Skew [58]

Radial Distortion

Radial distortion occurs when the light rays focus more on the edges of the lens instead of focusing in the optical center. For smaller lens, this distortion will be more accentuate. This type of distortion makes the straight lines in the world look curved in the 2D image. Radial distortion can be positive or negative. This effects are shown in Figure 16.

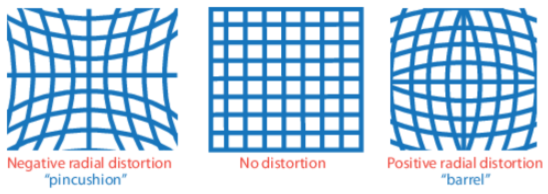


Figure 16: Types of Radial Distortion [58]

Also, the distorted points of Radial distortion are given by the following expressions, where x and y are the undistorted pixel locations, k1, k2 and k3 the radial distortion coefficients of the camera lens and $r^2 = x^2 + y^2$.

$$x_{distorted} = x * (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

$$y_{distorted} = y * (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

Tangential Distortion

Tangential Distortion occurs when the Lens of the camera and the image plane are not exactly parallel. This result into some parts of the image that look nearer than the others.

In Figure 17 it is possible to see how tangential distortion occurs.

Tangential Distortion points are given by the following expression

$$x_{distorted} = x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)]$$

$$y_{distorted} = y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y]$$

In this case the x, y, and r represent the same as the radial distortion expression, and p1 and p2 are the tangential distortion coefficients.

In this case, as the candidate photo was already submitted, it was considered that the camera calibration only used the following parameters: focal length of the camera, the optical

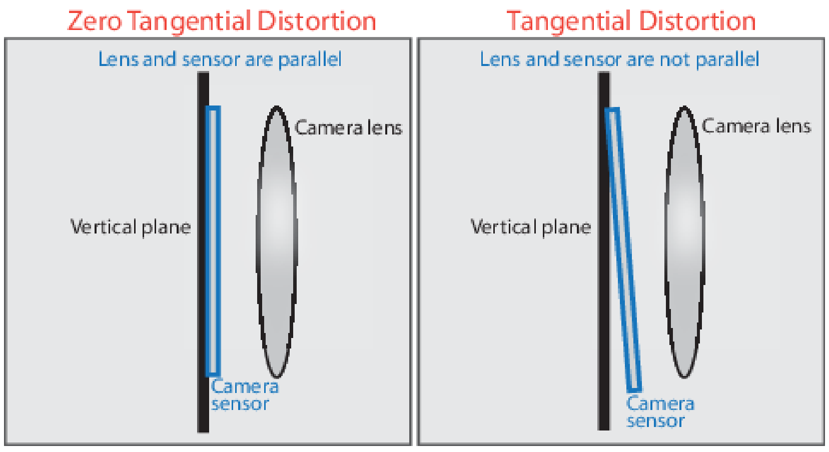


Figure 17: Tangential Distortion[58]

center and the radial distortion. This is a simple 3D model because it was not needed a really accurate one. So, the optical center was approximated to the center of the candidate image, the focal length to the width of the image and, finally, it was not used radial distortion at all.

The rotation and translation vectors were obtained using a `OpenCV` function, named `solvePnP`, that implements a lot of pose estimation algorithms depending on the given input argument in the flag parameter. In this case, was used a `SOLVEPNP_ITERATIVE` meaning that the pose estimation was computed based on Direct Linear Transform (DLT) followed by Levenberg-Marquardt optimization. DLT is a very common method that is applied to solve "almost" linear algebraic equations, specially when the objective is finding the relation between the projections points of a image plane and the respective 3D world points. In addition Levenberg-Marquardt Optimization is a method that allows to improve the DLT, because DLT is not very accurate and sometimes gives a high reprojection error. Basically this optimization changes slightly the pose of the object, in this case Rotation and translation vectors, and sees for which values it has the minimum reprojection error.

After getting the Rotation and translation vectors, it is applied a `OpenCV` function named `Rodrigues` [62] that allows to transform the Rotation vector into a Rotation matrix and with that it can be compute the projection matrix.

Finally, with the projection matrix it is obtained the Euler angles using the `OpenCV` function named `decomposeProjectionMatrix` [63]. If Roll, Pitch and Yaw were lower than 20 then it means that the candidate in the submitted photo has got a Frontal Face.

Sunglasses Detection

Another important criteria is to check if the candidate has got sunglasses in the submitted photo. Well, it is already implement a really trivial way to check this, that is simply compare the region below the eyes with the skin color, in this case with the tip of the nose, using HSV colormap, and if the colors matches using a defined threshold than the candidate has got no sunglasses.

So, looking again to the Landmarks on Figure 9, it was used the left eye lower points, 41 and 42, the right lower points, 47 and 48 and the nose tip point 31. After, it was calculated the distance between the two lower points of each eye, and it was add that distance to the left point of each eye, in this case the point will smaller x coordinate, in order to get the top left and bottom right points of the region below of each eye. Finally, it was split each region into the 3 channels of h,s,v and it was made the average of both regions for the S and V value. Then, it as subtracted to the skin reference the S and V average values in order to obtained to S and V differences. In this case the threshold used was for S and V difference values were lower than 90, then the candidate did not have glasses. To note that the image is converted to HSV color before comparing the regions mention in this text. This function can easily be improved by using a feature descriptor combined with a ML algorithm, for example.

Eyes Open

The Eyes in the photo can be open or close. There is a trivial way to check this through a formula named EAR that is in the expression below.

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

The p1, p2, p3, p4, p5 and p6 are 6 points corresponding to the eye. They are illustrate on Figure 18. Coming back to the landmarks Figure 9, for left eye, p1 corresponds to the 37 point, p2 to 38, p3 to 39, p4 to 40, p5 to 41 and p6 to 42 point. The same happens to the right eye, i.e, p1 is the 43, p2 is the 44, p3 is the 45, p4 is the 46, p5 is the 47 and p6 is the 48 point.

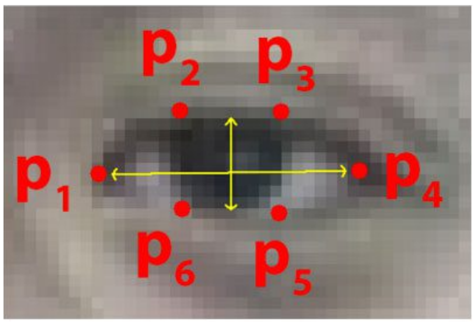


Figure 18: Eye Aspect Ratio Formula [64]

After applying the EAR formula it was made an average between the EAR of both eyes. Again, using a default threshold, if the EAR average is more than 0.2, than the candidate has its eyes open, otherwise they are closed.

Gaze

Now that is known if the candidate has got the eyes open, it is useful to see where is the candidate looking, i.e., computing the gaze. In order to do that, it is needed to detect the eye pupils.

First of all, is chosen a 3x3 Kernel, in this case it is used for morphological operations.

Kernels in Image Processing

In image processing a kernel [65] [66] is a convolution matrix used to modify an image. This kernel can represent a blur, a sharpening, a mask, and other types of filtering that are applied to the image. The expression to compute the new image through convolution is shown below, where g(x,y) is the new filtered image, f(x,y) is the original image and w is the kernel, i.e., the filter.

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x+dx, y+dy)$$

In order to get a better idea of how this kernel works, let´s have a look into the expression shown below that represents a kernel of a Box Blur, that is a linear box filter, where, every pixel of the image is calculated using the average of its value and the surrounding neighbours pixels. In this case, as it is possible to see, it is a 3x3 kernel, so every pixel has 9 neighbours, and behind the matrix is a factor scale of 1/9 that is multiplied with the sum off all pixel values.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

After choosing the adequate kernel, it is delimited the eye regions, i.e., the top left and bottom right coordinates of each eye, using again the Landmarks Figure 9. In this case for the left eye was used the 37 point for the x coordinate and the average of the 37 and 38 point for the y coordinate of the top left point and for the bottom right the x coordinate was based on the 40 point and the y coordinate on the average between the 40 and 41 point. The same process goes for the right eye.

With each eye coordinates region computed, first each eye region was converted into grayscale and, after, applied some filterings and contours to get the pupils. In this case, it was used a bilateral filter that is a filter that preserves the edges of the image and smooths the image. After it was applied a erosion, that is a morphological transformation that erodes the boundaries of the foreground object, in this case the pupil. Finally, it is applied a inverted binary treshold followed by the Otsu´s Binarization. The inverted binary threshold as the name suggest, invert the color regions, the pupil becomes white, and the Otsu´s Binarization serves to choose the best threshold value automatically. Then, is just applies a function that find the contours of each eye image and, "voilá", the pupils of each eye are detected. To get the focus for both eyes, it is computed the Centroid of each eye using the moments of the contours and then averaging the 2 values obtained to get the focus. As in the previous criteria, there is a specific threshold to know if the gaze of the candidate is pointing straight ahead. The best value found was 70, i.e., with a focus value higher than 70 it is considered a valid image.

Brightness

In order to check the image brightness it is converted the candidate image to a HSV color image and then it is simply made the split of the 3 image channels, H, S, and V and computed the average V value. This average gives the Brightness of the Image, as it was said in the Color Section, and the image is considered valid, with sufficient brightness if the value is more than 80.

Image Quality

The final criteria that is implemented is the image quality. Image quality can be classify using a final score of the image through IQA. IQA´s can be of 3 types:

- Full-Reference IQA - in this IQA, as the name suggest, the algorithm as got the original image and the distorted image and makes a comparison between them in order to get the final score.
- Reduced-Reference IQA - here the algorithm has got a little of information about the original image, for example a watermarked image, and the distorted image.
- Objective Blind or No-Reference IQA - this IQA only have the distorted image to qualify.

In the current software it is implemented a Objective Blind or No-Reference IQA algorithm, that is Blind/Referenceless Image Spatial Quality Evaluator or more common mention BRISQUE. One important note to take into consideration, it is the criteria to be a distorted image or the original image. In this case, is considered that the original (natural) image is an image that is directly captured by the camera and does not have post processing at all, while the distorted image has got some sort of post processing like a blur for example. However, it is hard for a algorithm tell if an image is natural or not, because with good filters the distorted image has got good quality too, so in order to get the final score it is used the Mean Quality score, that is a score that was made for academic purposes with many humans opinions and a dataset named “TID2008”.

The Mean Quality score of the image goes from 0 to 100, meaning 0 the best value and 100 the worst. This is possible to see in Figure 19.

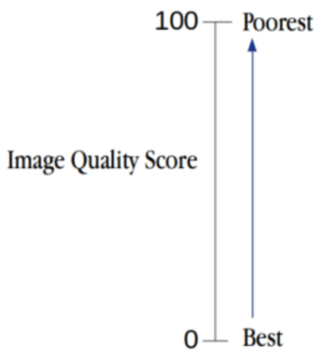


Figure 19: Image Quality Score [33]

BRISQUE is an algorithm that used the previous score and is divided into 3 steps:

- Extract Natural Scene Statistics (NSS)
- Compute the Feature Vectors
- Prediction of Image Quality Score (IQS)

In order to Extract the NSS it is needed to see the normalized pixel intensities curve. For an natural image this curve tends to be a Bell curve, i.e, a curve with Gaussian distribution, while the distorted image tends to have a deviation from that curve that tells the amount of distortion that that image has.

To normalized the image it can be used any sort of normalization algorithm. In this case it is used a Mean Substracted Contrast Normalization (MSCN), that basically, subtracts to the original image the Local Mean Field μ and then divide the result by the Local variance Field usually represented with a sigma. This normalizes the pixel intensity giving as the final result the luminance of each pixel. The expression that allows to normalized the image is represented below, where the $I(i,j)$ is the image intensity at pixel (i,j), the $\mu(i,j)$ is the local mean field and $\sigma(i,j)$ is the local variance field. Basically, the local mean field is the Gaussian Blur of the image and the local variance field is the Gaussian Blur of the square of the difference of original image and the local mean field. The expressions that computes the μ and σ are shown below, where W represents the Gaussian Blur Window function.

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C}$$

$$\mu = W * I$$

$$\sigma = \sqrt{W * (I - \mu)^2}$$

Although this is a simple and good way of normalize the image, the difference between the original and distorted image does not only resides in the pixel intensity, but it also depends in the relation between the pixel and it´s surroundings pixels,i.e, the neighbors. In order to get this relation, it is computed a pair-wise products of MSCN image with a shifted version of the MSCN image. This pair-wise products are represented in the following expressions, where H is the Horizontal product, V the Vertical and D1 and D2 the Left-Diagonal and Right-Diagonal, resulting into 4 new images for each one of the products.

$$H(i,j) = \hat{I}(i,j)\hat{I}(i,j+1)$$

$$V(i,j) = \hat{I}(i,j)\hat{I}(i+1,j)$$

$$D1(i,j) = \hat{I}(i,j)\hat{I}(i+1,j+1)$$

$$D2(i,j) = \hat{I}(i,j)\hat{I}(i+1,j-1)$$

To calculate the Feature Vectors, it will be used the 5 images that are computed so far. Basically the feature vector is always a 36x1 vector that is independent of the image resolution.

This vector is composed by the Generalized Gaussian Distribution (GGD) of the original image, which has 2 parameters that are the shape and the variance, and are located in the first 2 elements of the vector, and the Asymmetric GGD of the pair-wise product images, that has got 4 parameters, the shape, mean, left variance and right variance and these are located in the next 16 parameters of the vector since it was computed 4 different product images. The next 18 elements of the feature vector are filled with the same parameters that are calculated again, because the first ones were computed using only half of the image resolution and now they will be computed using the original size images.

Finally, it is made the prediction of the IQS using a ML model, for example a support vector machine.

In the implemented function all the description above was abstracted and used a BRISQUE model provided by the `OpenCV` library.

System Interconnection - FotoFaces APP

In this section the work done by 5 students from the UA Computing course of their 3rd year of graduation to a subject named "Industrial Project" is described.

These students created an mobile phone API that runs the FotoFaces software developed in this dissertation. The architecture of this system is composed by 3 main layers, namely: FotoFaces API, Mobile App and Database. This architecture can be observed in Figure 20

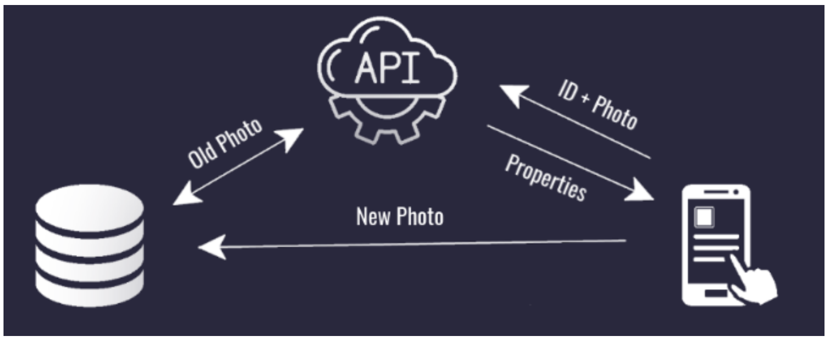


Figure 20: FotoFaces Architecture

In short, the API has the software algorithms developed in this dissertation as well as the previous ones that were already implemented and did not have any change. The database contains all the information used by its users. Finally, the Mobile App has some other algorithms to not overload the FotoFaces API, like a live detection algorithm that was implemented by the 5 students.

In terms of operation, the FotoFaces API waits for a message from the Mobile APP with the user submited photo, the user candidate name and its Identifier (ID). When the message is received, the API checks the user ID and accesses the database to search for the old photo and all the information about that user. After that, the API analyses the photo and sends the values (properties) of the photo to the Mobile APP. Finally, the Mobile APP proceeds to evaluate the photo using their custom criteria. In case that the photo passes all the criteria, then the photo is updated in the database. To note that this is a plugin architecture, which means that the system can be changed and adapted to the needs of the company that uses it.

The user interface of the Mobile APP is shown in Figures 21, 22, 23 and 24.

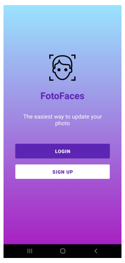


Figure 21: Start screen Interface

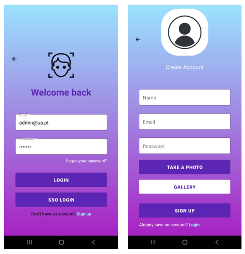


Figure 22: Login and Register screen Interface

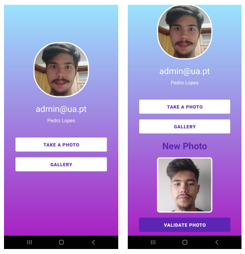


Figure 23: Main and Update photo screen Interface

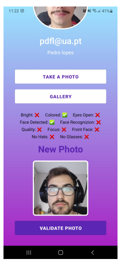


Figure 24: Validation Results screeen Interface

To not overextend this section more details about this System can be found in the article of this project.

Repositories

All the source code used to develop this system and the articles referred during the text ,in this case the Project 1 and 2 of “Foundations of Machine Learning” can be found here: `https://github.com/AntonioGoncalves99/FotoFaces`. Also, it is possible to find in this github repository a `Readme.txt` file that explains and give more details about all files and directories that are available.