The University of Manchester
Department of Computer Science
Project Report 2021-2022

# Exploring Face Detection, Face Recognition and Facial Expression Recognition

Author: Horia-Gabriel Radu

Supervisor: Dr. Tim Morris

Date: 29 April 2022

## Abstract

### Exploring Face Detection, Face Recognition and Facial Expression Recognition

Author: Horia-Gabriel Radu

Supervisor: Dr. Tim Morris

In this paper I explore Face Detection, Face Recognition and Facial Expression Recognition in images and videos. I explain the history of Object and Face Detection from older Classic methods like Viola-Jones, to recents Deep Learning methods like YOLO. Then I compare various methods to see how they function, what improvements have been made and what improvements are still in the making. I do the same for Face Recognition For Facial Expression and Emotion Recognition I perform Transfer Learning on two pre-trained Neural Networks and demonstrate how we can fine tune a model. Thus, I show how to implement a Deep Learning approach and use it for any task we want. I managed all of this by creating a C++ code base for the implementation of some methods, together with Python for Deep Learning and their analysis. This also shows the integration capabilities of each approach into multiple systems and devices. I evaluate each method for each explored task using my program and concluded that Deep Learning methods perform significantly better than Classic methods. Also, I highlight the data hungriness for the training, evaluation and testing of those Deep Learning methods, how they can impact your models and possible ways to solve this.

Overall, I try to give relevant information that would enable anyone to decide on what and how to use not only in Face Detection, Face Recognition and Facial Expression Recognition, but also in Object Detection and Classification.

# Contents


**1 Introduction** **3**
- 1.1 Reasoning . . . 3
- 1.2 Motivation . . . 4

**2 Methodology** **5**
- 2.1 Background . . . 5
  - 2.1.1 Traditional Face Detection . . . 5
  - 2.1.2 Face Detection based on Deep Learning Methods . . . 6
  - 2.1.3 Human and Facial Recognition . . . 8
  - 2.1.4 Facial Expression / Emotion Recognition . . . 11
- 2.2 Implemented Methods . . . 11
- 2.3 Techniques . . . 12
  - 2.3.1 Used languages for the code . . . 12
  - 2.3.2 Libraries, frameworks and helpful material . . . 13

**3 Approaches** **14**
- 3.1 Face Detection . . . 14
  - 3.1.1 Haar Cascade Frontal Face Detector . . . 14
  - 3.1.2 Histogram of Oriented Gradients + SVM Face Detector . . . 17
  - 3.1.3 Maximum Margin Object Detector (MMOD) CNN Face Detector . . . 19
  - 3.1.4 DNN-SSD (ResNet) Caffe Face Detector . . . 19
  - 3.1.5 OpenCV DNN-SSD Tensorflow Face Detector . . . 21
  - 3.1.6 You Only Look Once v3 Neural Network (YOLOv3) Face Detector . . . 21
- 3.2 Facial Recognition . . . 26
  - 3.2.1 Eigenfaces . . . 26
  - 3.2.2 Fisherfaces . . . 27
  - 3.2.3 Local Binary Patterns Histograms (LBPH) . . . 28
  - 3.2.4 DLib ResNet Neural Network Face Recognition model . . . 31
- 3.3 Emotion Detection . . . 33
  - 3.3.1 EfficientNet . . . 34
  - 3.3.2 Residual Neural Network (ResNet) . . . 34

**4 Results** **37**
- 4.1 Application and Program . . . 37
- 4.2 Evaluation and Comparison . . . 37
  - 4.2.1 Face Detection . . . 37
  - 4.2.2 Facial Recognition . . . 42
  - 4.2.3 Emotion Detection . . . 43

**5 Conclusion** **47**
5.1 Results Discussion . . . . . 47
5.2 Ethical Consideration . . . . . 48
5.3 Reflection and Future Work . . . . . 48

**Bibliography** **49**

**Appendices** **56**

# Chapter 1

# Introduction

## 1.1 Reasoning

We are surrounded by technology meant to understand images and videos, but how do these systems work, and what are their limitations at the moment? How do we see these technologies having an impact in the years to come.

Image Processing, Computer Vision and Artificial Intelligence are important research areas with many applications and a complex and evolving domain. For example, image processing can be used to improve the quality of images, computer vision can be used to automatically identify objects in images, while AI can be used in making a machine try to understand an image and make a decision based on the data received from it.

There are already widespread apps using those technologies, like the Face ID that Apple has and face recognizer for automated border control systems at airports.

Many of the most successful companies are also those that have built their businesses on the back of Image Processing and Computer Vision algorithms, or that have integrated them into their ideas, adding AI as a support for them.

Consider Facebook, which uses face recognition to suggest tags for photos; Google, which uses street view images to map the world; or Tesla, which makes use of Artificial Intelligence to make their self-driving vehicles possible. These are just three examples of the many ways in which Image Processing, Computer Vision and AI are used in the real world.

Furthermore, emotion and facial expression recognition has a huge potential to be used in gathering data, reviews, and experiences from a vast number of people in various situations, which could lead to major improvements in User Experience and rating of different products and activities. A lot of times expression and mood changes can be subtle to the human eye, as they can last for as little as half a second, but those micro expressions can mean a lot to what each human is feeling. So, we can make use of assisting machines to help us precisely identify those sudden emotion changes.

If we consider the saying **"A picture is worth a thousand words"** true, then by making a computer or machine understand all those a thousand words correctly, we are going to make our lives a lot easier and start developing technologies for the future. Of course that apart from Image Processing and Computer Vision, I also believe that Artificial Intelligence will be a great contributing factor in this endeavour.

## 1.2 Motivation

I have always been fascinated about computers and making them "understand" photos or frames, about Image Processing, Computer Vision and how Machine Learning, Deep Learning and Artificial Intelligence can help influence and improve them.

This technology that surrounds us is constantly improving and becoming more sophisticated. Systems made to try to understand images and videos are constantly improving as well. However, there are still limitations to these systems that we have to overcome.

I also see a lot of potential for new applications in those areas. For example, medical images could be processed to automatically detect tumours, they could have a use in web scraping for certain people or famous personalities online, security cameras could be made to automatically identify suspicious activity, intruders, or possible unlawful actions in traffic, or it can be added to a smart doorbell camera for recognizing visitors. The possibilities are endless, and I believe that there is a lot of room for innovation in this area, which is going to develop at a fast pace in the following years.

As such, I focused on exploring current algorithms and models, learning how they work, compare them, and how they can be combined into making applications to a greater extent.

The background of all three of the ideas I have tackled in this project, intertwine to form a complete picture of how they could all work together and be integrated into current technologies.The uses range from a robot that detects rubbish on the ground or in the sea and collects it; up to a scientific space robot that has to collect data from a planet, following a routine and making decisions on its own.

Identifying an object, recognizing the object and then trying to form an opinion based on the state of the object or the environment and act according to it, is basically how a human acts. So, exploring this was a way for me to better understand AI (Neural Networks), Computer Vision and their integration into any type of machine, albeit CCTV cameras or robots.

# Chapter 2

# Methodology

The field of Computer Vision together Machine Learning and Deep Learning is very vast, creating and training systems to capture, process and interpret data and information from either images or videos, can be a fascinating but tedious task.

For this study I decided on studying human faces, the difference between people's faces, recognizing key points and comparing them, and guessing their expressions and emotions; in both images and videos.

## 2.1 Background

### 2.1.1 Traditional Face Detection

The most know about and one of the earlier research done into face detection was in 2001, when Paul Viola and Michael Jones published a paper titled "Robust Real-Time Face Detection".

This paper explained in detail a method for detecting faces in images that was capable of running in real time on standard hardware at that time. While this algorithm had its limitations, it was, and still is, widely credited as being what sparked the interest of the computer vision community regarding face detection.

This proposed detection framework, which ended up being called **Viola-Jones** after its two researchers, and although it was initially trained to detect a variety of objects, its creation was primarily motivated by the problem of face detection [vio, 2022].

This algorithm works based on a sliding window which searches for **Haar-like features** in images and videos. Haar-like features are a type of feature used in computer vision for object detection, a type of wavelet used in signal and image compression. The algorithm also used Detection Cascade classifiers to reduce extra non-needed computation for the background of images.

All of the above together with the performance improvement from the Adaptive Boosting Training algorithm (AdaBoost) became very popular in the object and human detection area, seeing uses to this day.

A few years later, in 2005, N. Dalal and B. Triggs proposed another algorithm for detection called **Histogram of Oriented Gradients** [Dalal and Triggs, 2005]. They have stated that HOG descriptors outperform feature sets of key points for human detection, basically algorithms based on or similar to Viola-Jones, that use Haar wavelets. This was also proven with more research by multiple people [Rahmad et al., 2020], that HOG has better accuracy, less False Positives, performs much better on scale invariant input images and has fewer problems

with occlusion, clutter, and noise; while also theoretically being faster and less computational intensive.

HOG works based on blocks, very similar to the sliding window from Viola-Jones, dense grids on the image in which gradients are constituted from the magnitude and direction of change in the intensities of the pixels within each grid [med, 2022e]. It also has a rescaling factor for the input image, which is why it has fewer problems with its detection on various scale of the object.

**Local binary patterns (LBP)** are visual descriptors used for classification of features in Computer Vision. They were initially proposed in 1990 but have seen use later as it was proven in 2009 and 2015 that they significantly improve performance and accuracy of HOG descriptors [Wang et al., 2009] [Silva et al., 2015]. This goes on to prove that improvement are being made even years later to these well known algorithms.

**Scale Invariant Feature Transform** is also another Computer Vision detection algorithm, but it leans more over to the idea of being a matching classifier, as it is trained to detect and compare key points of objects from a set of reference images, in input images, based on the Euclidean distance of their feature vectors. This algorithm also makes use of a generalization of Hough transformation voting for filtering out good matches from bad ones.

Whilst those methods were sufficient at that time, having a satisfactory precision and speed, more and more limitations started to pile up. These limitations range from the face being too small in the image, it being oriented at a very harsh angle, or only parts of the face being visible. As an example, faces on a CCTV camera which can prove to be very small or faces covered by face masks.

Another very important aspect was the speed of these methods, how well they worked in real time, and their hardware requirements, so that they can be publicly distributed to numerous systems worldwide and be put in use.

### 2.1.2 Face Detection based on Deep Learning Methods

Object detection in Computer Vision then saw a rebirth with the introduction of deep learning based methods, precisely Convolutional Neural Networks.

Since the early 1950s, computer scientists have been trying to make use of Artificial Intelligence in understanding visual data [Hazarika and Kumari, 2019]. There have been highs and lows up until the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was established in 2010 [Zhao et al., 2019].

The first two years introduced only small improvements for Computer Vision together with the Artificial Intelligence field, but in 2012, a team of researchers implemented a **Deep Convolutional Neural Network** (DCNN) that surpassed by a wide margin state-of-the-art implementations results up until that point in time [Krizhevsky et al., 2012]. It later came to be known as AlexNet.

**CNNs** are neural networks comprised of multiple layers of artificial neurons that handle very well information from visual image representations. Those CNNs can vary in speed and performance based on their number of layers, type of layers and training of the weights, and they can be used not only for human detection, but for any type of feature extracting, pattern recognition, matching or natural language processing and understanding.

Following that, Artificial Intelligence development and improvements have found immense success in the Computer Vision field [mot, 2022].

From precision improvements, the focus was then put on algorithmically improving processing speed of those neural network architectures, going from models such as Region-Based Convolutional Neural Network (R-CNN), to Spatial Pyramid Pooling Networks (SPPNet), Fast R-CNN and Faster R-CNN [Girshick, 2015] [Ren et al., 2015]. The enhancements were significant.

| | R-CNN | Fast R-CNN | Faster R-CNN |
|---|---|---|---|
| Test time per image | 47 seconds | 2 seconds | 0.2 seconds |
| Sped up | 1x | 23.5x | 235 |
| mAP(%) | 36.73 | 42.56 | 54.83 |

Table 2.1: Table from [Li, 2021] and [Ren et al., 2015], tested on PASCAL VOC2007, COCO and ILSVRC data sets with the same neural network setup.

More research was also done on detecting smaller objects in images. For this, Feature Pyramid Networks (FPN) were one of the solutions introduced. They scale inputs to various sizes and allow for deeper detection and understanding of images when combined with Fast R-CNN and Faster R-CNN.

There are still many more algorithms, implementations, and variations regarding object detection, and more are being researched and developed as this industry is still growing at a fast pace, but I will not go through all of them.

With that said, I will still mention 4 more methods that in my opinion have promising results and high potential:

- **You Only Look Once (YOLO) algorithm**, initially announced in 2015, is a one-step detection algorithm that has a very high optimization of the detection performance. It only looks at an image once to detect multiple objects, handling decently well even small objects.

  We will look more in depth into this algorithm further, as it is included in the ones I implemented. While it has the least accuracy of the 4, it is theoretically considered faster and preferred in certain scenarios where speed is key.

- **Single Shot (Multi-Box) Detector (SSD)** is another algorithm for quick and accurate object detection. Similar in a way to YOLO, it only takes one look to detect multiple objects present in an image. These predictions are received from multiple bounding boxes of fixed sizes, then they are scored by the presence of each desired object in each box, ultimately having a non-maximum suppression function produce the final results. It is most known for its introduction of multi reference and multi resolution detection techniques in one-step detection.

  Some papers state that SSD is faster than YOLO, and that is not entirely wrong, but because it relies on bounding boxes for making its predictions, increasing the size of the input image significantly lowers its speed. So, a SSD for 300×300 size is faster than YOLO for 300×300 size, but for a 500×500 input size, SSD significantly loses speed performance compared to YOLO [Kim et al., 2020] [Tan et al., 2021].

So, we always need to take in account the size of the input to compare speed performance. SSD is considered to not work very well on smaller objects but makes more accurate predictions for True Positives, having a better precision of intersection over union [Liu et al., 2016].

- **Retina Net** is also a one stage detector that makes use of a Focal loss function to combat class imbalance in the training phase of the neural network, basically a cross entropy loss function for optimization of the detector.

  While SSD and YOLO are fast, they tend to be less accurate than region proposal methods (two stage methods), so RetinaNet intends to alleviate this problem with the Focal loss and Feature Pyramid Network introduced into their architecture.

  Altogether, it has significantly better accuracy, even on scale invariant images, but with the added complexity it becomes more processing extensive and has a slower inference time for forward propagation [med, 2022f].

- **Detection Transformer (DETR)** is a new type of approach to the object detection field. It sees this challenge as a direct set prediction problem, while using a whole different architecture, a Transformer [Carion et al., 2020].

  The Transformer learns to predict collections of unordered data with unknown relations, and as such outputs a set of predictions based on the input image. This method is also known as End to End Object Detection using a Transformer encoder-decoder architecture, and it was release much later, in 2020 by Facebook AI. The results from the experiments carried out by the DETR researchers on the COCO dataset proved that it achieves comparable results to the widely used and successful Faster RCNN [Lin et al., 2014]. The researchers also state that there is still a lot of room for optimization regarding training and performance in future works [Carion et al., 2020].

  While it took years of development for different methods to compete with state-of-the-art performance, DETR achieved this in a much shorter period of time, considering the fairly recent introduction of this approach.

  With this example I would like to note how huge technological advancement and discoveries are still being made, and the impact deep learning has had, and it keeps on having in this industry, where different methods, scarcely thought about in the past, proved great concepts in already explored areas [tra, 2022a].

Face Detection is a very complicated problem. There are a lot of different people, younger and older, each person can have different feature, some can wear glasses or have longer hair that covers parts of their face. Then, there are also outside factors, like the lighting and shadows on a face, or the resolution and noise in the video or images used for the detection.

One important point to be made is that very rarely we get a method that performs well in all areas, speed, accuracy, hardware requirements, and handling visual limitations, noise, or occlusion. Most of the time, there is a trade-off between speed and accuracy. Developers have to compare the pros and cons of each method and make a choice based on what they need it for. This was very nicely compared and explained in an article by Jonathan Hui [med, 2022f].

### 2.1.3 Human and Facial Recognition

The earliest steps into face recognition in images were made back in 1964 and 1965 by Woody Bledsoe, Helen Chan Wolf and Charles Bisson, through manual feature extraction from

images. Because of their presumed work being funded by an unnamed intelligence and security agency, their research was never release to the public, but it has been considered a milestone for the potential that face recognition would be having in the future for biometrics and security.

Fast-forward years later and this area has been greatly developed and researched. Similarly to object detection, there are a number of ways in which you can approach human recognition and matching. These methods can be of multiple types, some very different from others: classic face recognition algorithms, face descriptors, feature vector based model; neural network architectures trained with deep learning; three-dimensional recognition of a person's head (skull); skin texture analysis; thermal cameras and infrared sensors; retina scans; pattern matching; and more types of frameworks, a lot of which are based on deep learning [Wójcik et al., 2016].

Moreover, they are frequently used together, for example how the Face ID from apple scans both the face from the frontal camera, but also captures an infrared image of your face to determine more specific data like depth of features or occlusion. There can be a whole discussion about biometrics and human recognition, but I will only approach Face Recognition in Computer Vision, data which is interpreted only from a face in images or videos.

Those methods can be roughly categorized as 5 main types [Trigueros et al., 2018]:

- **Geometry based Methods**
  This is considered to be the first type of approach used in the early systems for automatic face recognition. It is based on using edges to find locations of facial landmarks, and then calculate the distances and positions of those landmarks from each other, by using the Euclidean Distance.

  Those measures would then be compared with other measures from faces to make predictions with certain confidence scores.

  Even if this method proved to have a decent computation time and not need a lot of processing power, the accuracy was mediocre. But it proved to be a great step into face recognition as other hybrid methods emerged, that combined ideas from this method with more advanced ones, which provided much better results.

- **Holistic Methods**
  This method analyses the entire human face as one unit for face recognition. The faces are projected in a vector subspace to get rid of as many as possible superfluous factors and then analysed. Two of the most popular methods for this are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

  The PCA approach is also called Eigenfaces method, because of the resulting eigenvectors from the linear transformation in the subspace. The calculated weights from those vectors are then used to identify faces in input images.

  LDA method was proposed to improve the accuracy of the PCA, due to variance issues that could arise in the latter, such as illumination, rotation, or occlusion. Sometimes called Fisherfaces method, LDA uses class labels to find the projection that maximizes variance between classes and minimizes variance in classes [Trigueros et al., 2018]. A combination of the 2 methods proved to be the best option, as LDA was prone to overfitting due to smaller number of classes.

  Support vector machines (SVM) have also seen usage in the face recognition holistic methods, in the comparison and distinction of face labels. SVMs are supervised machine learning methods used for classification and regression tasks.

Holistic methods showed overall greater accuracy results than Geometric based methods, albeit Geometric methods were usually faster.

- **Feature based Methods**
  They are somewhat based on the Geometry methods because they extract discriminative features, but they do not compute their geometry. These methods act better when dealing with faces that have different outside parameters, like illumination, expression, and rotation of the face. The vectors extracted from the face features are compared between each other and predictions are made. Those key points are usually based on the: eyes, nose, mouth, face shape, chin and eyebrows, from which descriptors for several classes are computed.

  Feature based methods are extensions of different object detection algorithms that capture each facial landmark, plus algorithms used for transformation and comparison of the weight representation of the obtained vectors. Examples include histogram orientation descriptors calculations used for this type of recognition [Freeman and Roth, 1995], as well as Local Binary Pattern descriptors together with feature extractions from Gabor filtering transformation mapping, or Scale Invariant Feature Transformation algorithm. There is also an extension of the Eigenfaces method that produces sets of eigenfeatures instead of holistic eigenvectors to classify the face.

  Overall, feature based models improved handled edge cases fairly well, but were slower in comparison to Holistic methods.

- **Hybrid Methods**
  This approach combines both Holistic and Feature based methods for a deeper exploration of the face, taking the performance and robustness from both methods. Before Deep Learning methods were largely popularized, those methods were considered the state of the art in face recognition.

  The most popular hybrid approach is to use LBP or SIFT for extracting local features from a face and then project them into a discriminative and low dimension subspace using PCA or LDA [Trigueros et al., 2018].

  A lot of research has gone into mixing and matching different Holistic and Feature based methods in an attempt to get rid of as many limitations as possible and get the best results and performance. A main complexity in Hybrid and Feature based method comes from how to decide which features are the most useful for recognizing and comparing faces. This is where deep learning methods made an impact.

- **Deep Learning Methods**
  As the name implies, Deep Learning Methods makes use of neural networks to solve the face recognition task. Deep Learning Neural Networks turned out to be satisfactory approaches due to them being able to learn what are the optimal features to use for key points extraction and what is the best method to use in representing and comparing them.

  Convolutional neural networks are the most widely used type of neural networks for facial recognition. There are three components to take in consideration when using CNNs: the neural network's architecture, the loss functions used, and the data used for training.

  Those architectures and loss functions are comparable to the ones used for object detection, the main difference being their use case and how the weights of the model are trained. Object detection CNNs try to find known objects and similar ones in images, and face recognition CNNs look for similar features between a number of faces. Usually for

speed purposes, those descriptors are serialized or vectorized into a known set subspace, and then compared.

Also, the main drawbacks, as is in most Deep Learning methods, are the need for large annotated data set for expensive training and the ambiguity induced by a Neural Network, not being able to fully control how the model perceives and classifies the data.

Out of all 5 methods, Deep Learning is the most widespread in the present, because, provided you have enough data to train your models, you will achieve the best overall results. This is also the state-of-the-art method for face recognition, with Facebook's DeepFace neural network reaching an accuracy of 97.35% on the Labelled Faces in the Wild (LFW) dataset [Taigman et al., 2014]. It reduced the error rate of the previous state-of-the-art method by 27%, on the same dataset.

### 2.1.4 Facial Expression / Emotion Recognition

When it comes to Facial Expression or Emotion Recognition, we talk about a classification task, simply put, assigning objects to classes. The classification can be solved using:

- Simple feature extraction and comparison methods, such as Histogram of Oriented Gradients (HOG)
- More complex feature detection and classification methods that make use of machine learning. There are a number of them, but a few examples would be Support Vector Machines, Logistic Regression, Naive Bayes models or Hidden Markov Models based implementations.
- Multi Label Classification with Deep Learning methods, based on Artificial Neural Networks. Because of usually having a set number of emotion classes that the Neural Network has to learn, there is are a huge number of architectures that can be used: CNNs, Multi Layer Perceptrons, Long Short Term Memory, et al.

## 2.2 Implemented Methods

Out of all the listed methods of the three tasks enumerated above, I have chosen a few to deeply study, implement and evaluate.

**Face Detection** For this, I wanted to explore methods from traditional object detection to Deep Learning approaches for object detection to state-of-the-art algorithms made in recent years (YOLOv3).

1. Haar Cascade Frontal Face Detector which uses the Viola Jones face detection algorithm
2. Histogram of Oriented Gradient Face Detector that uses Support Vector Machines
3. Convolutional Neural Network based method that uses Maximum Margin Object Detection (MMOD), trained for Face Detection
4. Deep Neural Network approach based on a Single Shot Multi-box Detector (SSD) made using Caffe
5. OpenCV own implemented Deep Neural Network Face Detector made using TensorFlow

6. You Only Look Once Version 3, a one stage object detection algorithm with trained weights for Face Detection

**Face Recognition** Here I wanted to discover various image processing, transformation, characterization and classification algorithms, but also compare them with state-of-the-art approaches such as Neural Network Models.

1. Eigenfaces
2. Fisherfaces
3. Local Binary Patterns Histograms (LBPH)
4. Face Recognition model from DLib, a Deep Learning approach which uses a ResNet Neural Network

**Facial Expression/Emotion Recognition** And finally, I wanted to create my own Deep Learning model for classification with an already made Neural Network architecture. I fine-tuned 2 architectures:

1. EfficientNet, a Convolutional Neural Network
2. Residual Neural Network (ResNet), an Artificial Neural Network

By delving deeper into those specified methods for each task, I plan to learn and present how they work. I also want to highlight the evolution of object detection and classification in Computer Vision and what impact Machine Learning, and Deep Learning, has had on it. Most importantly, I would understand each algorithm, limitation, possible workaround, solution, and technicality for those methods better.

Finally, I evaluate and compare each one of them to form an opinion on what is already available in the industry, what is state of the art, and get insight of the potential this field can have in the future.

## 2.3 Techniques

Of course, for this project, I needed to decide how I would implement it. I tried to use well known and common techniques so that it would be easier to follow and because a larger number of well documented resources are available online.

### 2.3.1 Used languages for the code

I have used two programming languages for my project:

- **C++** for the implementation of each method in a program.

  There were a few reasons for me using C++.

  First, it runs much faster than a number of programming languages, Python for example, which would have been a strong alternative. This is because it is a pre-compiled language, very performant and memory efficient. While the compilation of the program takes time and resources, my main focus is ensuring the Face Detection and Recognition applications are able to run in real time and as fast as possible.

Second, it is an object-oriented language, very useful for the modular approach I took in writing the code so that it will be easier for me to compare each method, which has to be in the same environment with the same setting.

Third, the language is also very close to C, easily portable and compatible with various other languages.

Fourth, C and C++ are widely used for developing embedded systems, like smartwatches, phones, security cameras and vehicles, as they are closer to machine language/code. Object, face detection and recognition algorithms would be most used in this type of devices, plus servers, APIs, or applications.

Fifth, I wanted to get a deeper knowledge of C and C++ by undertaking this project, and learn more about how to better program in those languages.

Finally, I believe that by writing in C++, it would then be much easier to translate and port the code in other languages, so, overall, this is why I chose to write this exploration code base in C++ rather than Python.

- **Python** was also needed in this project.

  Implementing the Neural Network fine-tuning for the Emotion Recognition task is much more natural in Python.

  There are much more resources available for Machine Learning, Deep Learning and AI in Python, with many widespread frameworks like PyTorch, Keras, TensorFlow and so on.

  Furthermore, I chose it for the evaluation of my methods, as it is much easier to manipulate and work with big amounts of data and statistical information. Displaying and comparing results is also simpler when using Python and its available libraries and platforms (Matplotlib and Jupyter Notebook).

### 2.3.2 Libraries, frameworks and helpful material

I also needed help from certain frameworks and libraries to simplify the implementation. I decided on using very common and well known software and the most significant in my project are:

- **OpenCV** [ope, 2022] [lea, 2022] [Bradski, 2000]

  I mainly used it for image manipulation and visual processing of images and videos plus displaying. Its DNN module for Deep Neural Networks also proved very useful for accessing the Deep Learning approaches I implemented.

- **DLib** [dli, 2022] [King, 2009]

  It was used also in image manipulation and pre-processing, but I had access to different calculations from it that I wanted to compare with. DLib also contains Machine Learning tools and algorithms that come in handy when dealing with these type of tasks.

- **TensorFlow** [tf, 2022] [Abadi et al., 2015]

  It was used for the transfer learning and fine-tuning of the Neural Network architectures I trained for the Facial Expression Recognition classifiers. A good alternative was PyTorch, but I found Tensorflow worked better in combination with C/C++, for when I wanted to implement the resulting models into my C++ programs.

# Chapter 3

# Approaches

As I have described in the Background section, there are a lot of ways you can approach each task.

I could not have implemented all of them, so instead, I decided to pick out the most notable ones I could think of, that would also highlight the evolution of face and object detection, classification and recognition in Computer Vision.

I then evaluate and compare those, from classic methods like Viola Jones (Haar) to Deep Learning methods that make use of improvements from recent years.

## 3.1 Face Detection

Firstly, I will explore all the six Face Detection methods I implemented. I will talk about how they work, their architectures, frameworks, backbones, and their advantages and limitations.

### 3.1.1 Haar Cascade Frontal Face Detector

This method is based on the **Viola Jones algorithm**, from the paper "Rapid Object Detection using a Boosted Cascade of Simple Features" [Viola and Jones, 2001] proposed in 2001 by Paul Viola and Michael Jones.

Viola and Jones first had to deal with determining whether a bundle of pixels in an image is to be considered a face or not. For this, they used three kinds of features, reminiscent of Haar wavelets.

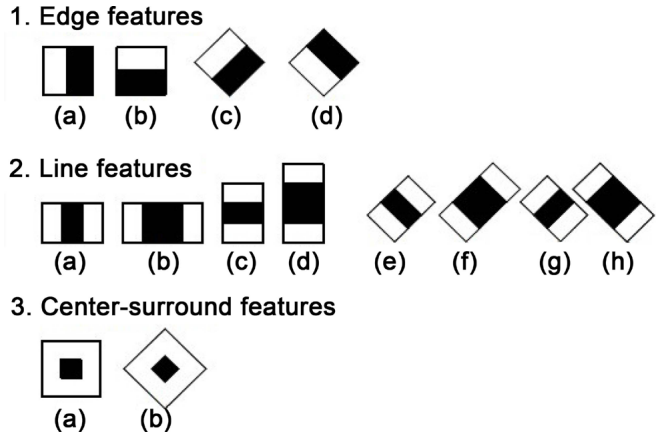



Figure 3.1: Haar-like feature types [Lienhart and Maydt, 2002] [Zhang et al., 2017].

Those **Haar-like features** are calculated by summing up the intensity of the pixels in each region (whiter and darker) and then taking the difference of these sums. This is done in the gray scale of the image.

These features would then be used to categorize subsections of a face, like the difference between the eyes and the cheeks, mouth and nose, or eyes and nose bridge, where some regions usually have lower or higher intensities of pixels between each other.

There is a problem with computing those differences though, they would take a lot of time, considering the detector is for a base resolution of 24×24 and the image can be much larger than that. The calculations for sum and differences of those areas would be very exhaustive on a normal gray scale matrix for the image, with each point in the matrix being a pixel.

So, the two researchers proposed a way to compute those features by representing the image as an **integral image**. The integral image contains the sum of the pixels above and to the left of each position x and y.

| | | | |
|---|---|---|---|
| 4 | 1 | 2 | 2 |
| 0 | 4 | 1 | 3 |
| 3 | 1 | 0 | 4 |
| 2 | 1 | 3 | 2 |

| | | | |
|---|---|---|---|
| 4 | 5 | 7 | 9 |
| 4 | 9 | 12 | 17 |
| 7 | 13 | 16 | 25 |
| 9 | 16 | 22 | 33 |

Figure 3.2: Image matrix of pixels represented as an integral image. Figure taken from [Bradley and Roth, 2007].

This representation allowed the computation to be done in very rapid and constant time, because each calculation would just be the sum and difference of 4 array references, no matter how big the group of pixels is.

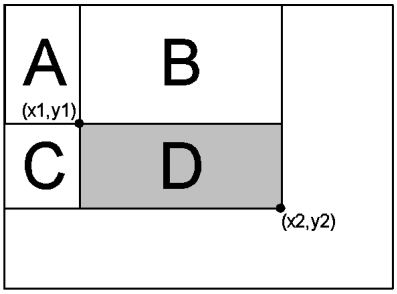



Figure 3.3: Sum of pixels in D square will be D + A - (B + C). This is how the integral image representation sped up calculations. Figure taken from [Bradley and Roth, 2007].

By combining the extraction of those features with the representation of the image in integral form, Paul and Viola formed a basis for how their method would work.

Next, they would need to determine which Haar features from a face would be best taken in consideration, to get as many True Positive and least False Positive predictions. The statistical method **AdaBoost**, short for Adaptive Boosting, was used as a learning algorithm to extract the

most meaningful features. Below is an example of the Haar-like features, the first and second ones selected by AdaBoost.

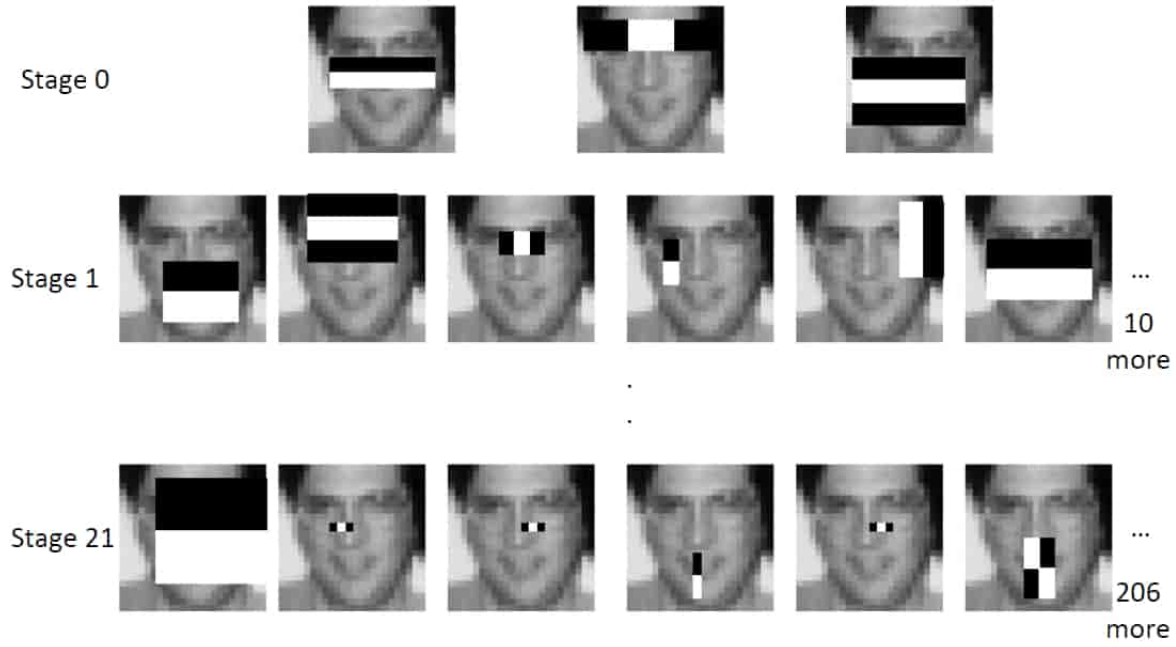



Figure 3.4: Stages of the Haar-like features and how they match on a face. Figure taken from [ima, 2022].

AdaBoost initially resulted 200 features for the frontal face classifier, which had a detection rate of 95% and a False Positive of 1 in 14084 [Viola and Jones, 2001]

Lastly, for an increased performance of the classifier, an Attention Cascade classifier was introduced. This radically reduced computation by arranging the evaluation of each feature in a series, where if an evaluation failed and provided a negative result (a Haar feature out of all was no present) the classifier would skip the sub window and go on to the next one. Basically, the Cascade classifier made sure to quickly go through the background of images where faces were clearly not present by rejecting from the start predictions in that specific region.

The feature screening process was divided into 38 layers, the total features in all layers counting up to 6061. The use of this technique can be noticed in the first five layers of the detector, with 1, 10, 25, 25 and 50 features respectively. The average number of features evaluated per sub window is 10 out of 6061. So, if a potential prediction fails at the first layers, it will not waste computation power to continue calculations, and by an increase in number of features from each layer to another, this was achieved successfully.

Methods similar to this one have seen a lot of use in object detection techniques. By pre-screening the image for possible matches and making deeper calculations only for possible faces, you would significantly increase computation time.

So, by improving the calculations of Haar-like features with integral images, determining the most meaningful features out of 160000 plus and applying a cascade classification on the predictions, Viola and Jones achieved a viable approach for object detection. This minimized the computation time and achieved high accuracy.

At that time, this was a revolutionary approach to face detection that performed at a decent speed, while having a satisfactory precision. This, plus its low hardware requirements, is why it is so widely used in simple face detection application and devices, a good example being the automatic focus in digital cameras.

This paper did not propose only ideas invented from scratch, as it can be seen in the references provided by their paper [Papageorgiou et al., 1998] [Rowley et al., 1998], but the way they combined each one of them to reform object detection proved very impactful in this area.

I wanted to present this approach in depth as the methods were and still are an inspiration to a good deal of research for Computer Vision, and the approach is still used to this day for this task, pre-processing of the image, extraction of the feature and speed and precision improvements.

OpenCV framework already contains a pre-trained Haar Cascade classifier for frontal faces in various size images, which is what I used in my project [Bradski, 2000]. The use of this Haar-like feature classifier comes with its limitations though:

- Occlusion, lighting and rotation of the face
- Faces of very small or big size

Because the face is determined based on the Haar features, the detector will struggle with a face rotated or covered right on the key features of the classifier. Lighting also plays an important part in detection because the Haar features are determined by the difference of the sum of intensities of pixels, from whiter to darker pixels. So, a normalization for this lighting could prove very significant.

Sometimes, a Histogram Equalization can help with normalizing the contrast in an image so that it is easier for the detector to evaluate those features. Also, it is harder to determine the feature of faces that are too small in large images, thus they will sometimes not pass the "threshold" of the detector.

My thoughts for this detector are that it is decent in simple devices like a digital camera, but it does not work very good in surveillance and CCTV cameras, as it struggles a bit with outside factors and small and rotated faces. I will explore and compare it further in the evaluation section.

### 3.1.2 Histogram of Oriented Gradients + SVM Face Detector

This method, introduced a few years later by Dalal and Triggs, is a different approach to human detection, using grids of histograms to determine descriptors for faces feature sets [Dalal and Triggs, 2005].

First, the image is split into small connected cells of the same size, depending on how big or small the object features you are trying to capture are. This also has an impact on the performance of the detector, more cells will result in a slower computing time but possibly higher precision.

Then an orientation histogram is calculated for each cell, determining each edge's orientation gradient in a local neighbourhood area.

The gradient orientation histogram has been used similarly in key points and edges detectors like Scale Invariant Feature Transformation or Canny Edge Detector

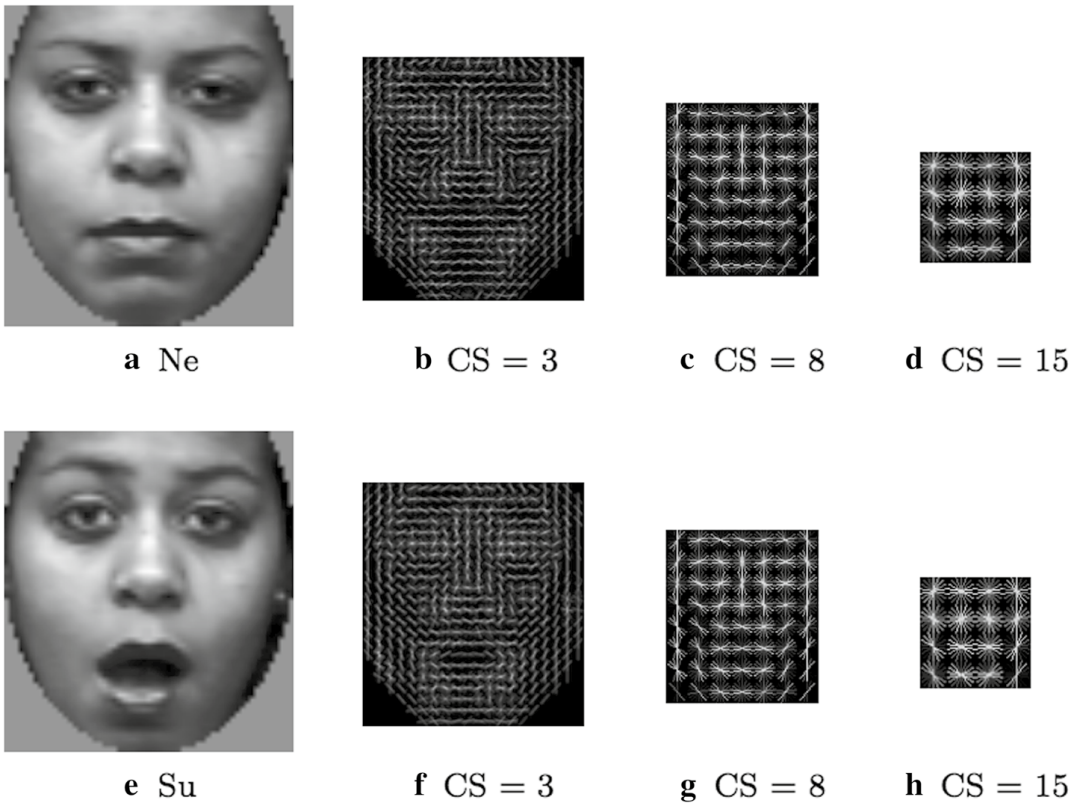



Figure 3.5: Examples of HOG images, where CS is the cell size of the processed image. Smaller cells extract more features and are more precise compared to less large ones. Figure taken from [Carcagnì et al., 2015].

Those orientation histogram gradients are used to predict and geometrically extract HOG descriptors from each cell in the image. They capture the edge or gradient structure, characterizing local shapes.

Finally, the groups of features in each cluster are then used to train a Support Vector Machines linear classification model. This algorithm was used because of its capability to work with high dimensional features [Dalal and Triggs, 2005].

The data from the histograms helps in the model learning various sets and sequences that have the highest probability of representing a face. Features of the face like eyes, mouth, nose, and eyebrows help the most as they represent the vast majority of edges found in a face.

The SVM model is then used to find cells in the image which contain a satisfactory number of descriptors over a certain threshold, so that those extracted cells contain a face.

A big advantage of this method was that it took a lot less time to train compared to Viola Jones algorithm, under 10 minute, while the latter detector took even days to train and learn the meaningful features. To keep in mind is that this was the case at the release of the research.

I implemented this method using the DLib library. The detector is a little slow because it has to compute a vast number of orientation histogram gradients for each image it receives and then use the Linear SVM to extract the face.

Because this method splits the input into a set number of cells, and makes predictions based on them, it struggles with smaller faces. Also, normalization is needed for this process due to

the fact that histograms are used, and intensive lighting and shadows can significantly lower the performance of the detector.

### 3.1.3 Maximum Margin Object Detector (MMOD) CNN Face Detector

This approach uses the **Max Margin Object Detection** presented by Davis E. King in [King, 2015]. The detector is a Convolutional Neural Network Model trained by the creator of DLib, the same person that presented this method, Davis E. King [King, 2009].

The presented method is algorithmically trained different, such that when training it takes a set of images and tried to find the best weight for which the detector makes the most correct prediction. A Max Margin procedure is then introduced so that the predictions are correctly made with a large margin. It works based on a sliding window classifier on the image, but through the complexity of the CNN it tries to reduce the number of unnecessary calculations it does for each sub window.

This is considered a Greedy Algorithm approach as it tries to find the best solution for the given training input images, and then retries the same approach for evaluation.

The Maximum Margin algorithm can be applied to other methods as well, to try to experiment if it improves them, as it was the case with the HOG filter concluded in [King, 2015]. It showed that using MM improves the True Positives versus False Positives rate of HOG.

While this approach might not be the best one, as I have noticed from my experiments, I found its ingenuity very refreshing. It goes to show that there may still be a lot of algorithms and methods that have not been discovered, that could be the next state of the art in this area of Computer Vision.

Overall, this approach is very easy to train as it does not need large amounts of training data, it handles occlusion and rotation decently well, and it has a good real time detection speed. The downside is that it stagnates in learning, and it is considerably hard to get it to current state-of-the-art performance without major changes or improvements, as it is a greedy solution to the problem.

### 3.1.4 DNN-SSD (ResNet) Caffe Face Detector

This Deep Learning approach was trained with the **Caffe** framework, created using a Single Shot Multi-box Detector framework and a Residual Artificial Neural Network (ResNet) backbone architecture.

CaffeNet was initially created by Yangqing Jia for his PhD at UC Berkeley, and further developed by Berkeley AI Research and some community contributors. It is a version of AlexNet, which was the 2012 winner of the ImageNet Large Scale Visual Recognition Competition (ILSVRC), that can be trained and simulated on just one GPU. The developers of CaffeNet were the ones that also developed Caffe, a deep learning framework, excelling in speed and modularity.

**ResNet** is an architecture that contains Rectified Linear Unit (RELU) activation functions and batch normalization methods. One of the uses of RELU is training visual feature extraction neural networks. Batch normalization is used to make each layer computation faster and more stable by re-scaling and re-centering the input data. ResNet avoids the problem of vanishing gradients by skipping connections of some layers, effectively simplifying the learning of the network in the initial stages. By solving the vanishing gradient problem, it is possible to then

use back propagation methods to perform supervised training of the deep neural network and maximize the learning.

There are a lot of ResNets available, ResNet-10, ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, each one being in the same family of CNN architectures, but having more Residual Units and layers as the number is higher. So, a higher number mean a bigger dimension and deeper complexity for training and evaluating, but also an increased time for the training and a slower feed forwarding to the network.

The specific architecture used for this model is a ResNet-10 for faster training and a real time performance in the detection stage.

The used SSD model framework, proposed in the research paper [Liu et al., 2016], is based on a feedforward Convolutional Neural Network with some added improvements.

Simply put, this method initially extracts feature maps from an image using the VGG16 neural network (Very deep convolutional network for large-scale image recognition) [Simonyan and Zisserman, 2014] as a base network; and then applies convolution filters to detect the desired objects, in this case, faces.

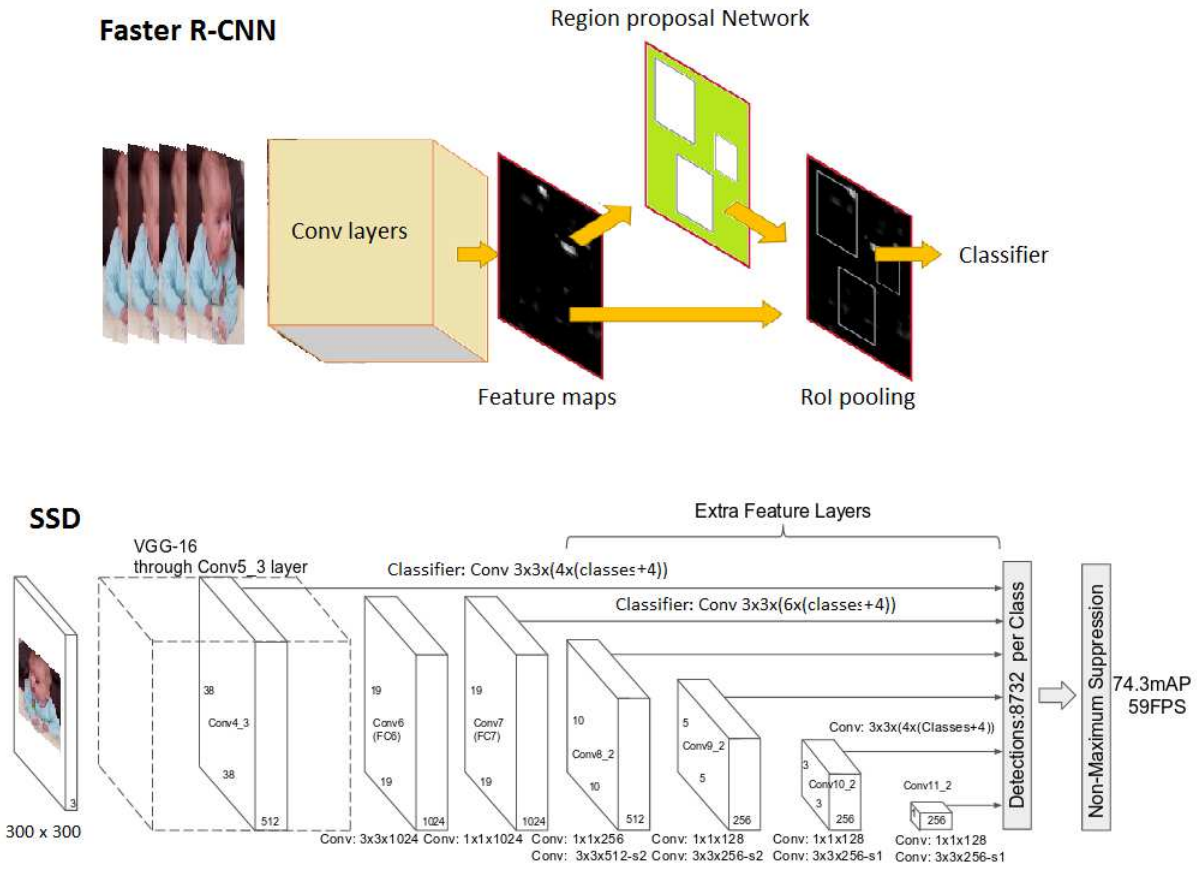



Figure 3.6: The SSD only has one stage detection in its architecture, compared to a normal R-CNN which has an additional stage (two stage detector). Figure is taken from [Phon-Amnuaisuk et al., 2018].

Considering this is a single shot detector, it only takes a single forward pass to the network for it to make the detection, resulting in speed improvements. While the emphasis is put on speed,

it still has a very good mean average precision in comparison with previous implementations such as Faster R-CNN.

Additionally, the multi bounding box regression technique, detailed in the paper [Erhan et al., 2014], significantly improves the accuracy for the intersection over union of the bounding boxes, making the detection much more fit onto the actual object or face.

The model I integrated in my project works on 300×300 input images, providing good frames per second in real time videos. It also deals very well with rotated or occluded faces. The main problem comes when the faces are too small for the detector, and because the model is trained on input of 300 by 300, it struggles in those cases.

### 3.1.5 OpenCV DNN-SSD Tensorflow Face Detector

This method is very similar to the DNN-SSD above, but it is trained and presented by the OpenCV DNN module using the Tensorflow framework. It uses the same Single Shot Multi Box Detector framework as a base and the Residual Neural Network architecture as backbone.

The difference is that the weight file used for detection is **quantized on an 8-bit unsigned integer channel** to significantly reduce the size of the previous model weights. The Caffe trained model is of size 5.4 MB, while this model is 2.7 MB, half the size. In this particular case the difference is not major, but for much bigger models the difference could be notable. The reducing of memory of a model weight can dictate whether an approach can be integrated into different devices and applications, that might not have enough space allocated for the big version of the model.

The use of the TensorFlow library for quantization should theoretically reduce CPU and hardware latency, processing power requirements and model size while having little impact on the accuracy. At inference, instead of using float points for the serialization of the protocol buffer file, binary format is used. This protobuf file holds the trained weights of the model.

This enhancement appeared very appealing as it could have a major impact on multiple approaches. So, by integrating this method, I want to survey if there would be any other major differences in how the model performs in detection precision and speed in my project.

### 3.1.6 You Only Look Once v3 Neural Network (YOLOv3) Face Detector

**YOLO** is a detection system considered state of the art in terms of speed performance, with precision and accuracy not lacking either.

The YOLO algorithm works by applying just a single neural network forward propagation, from the input layer to the output layer, to the whole image. Just as the name implies, you only look once, only a pass through the detector's layers is needed for the classification and localization of the objects or faces you want to detect.

This algorithm was initially presented in 2015 by Joseph Redmon et al. in the research paper [Redmon et al., 2016] as a new approach to object detection. Instead of tackling object detection as a classification task, they addressed it as a regression problem to box bounding and predicting class probabilities.

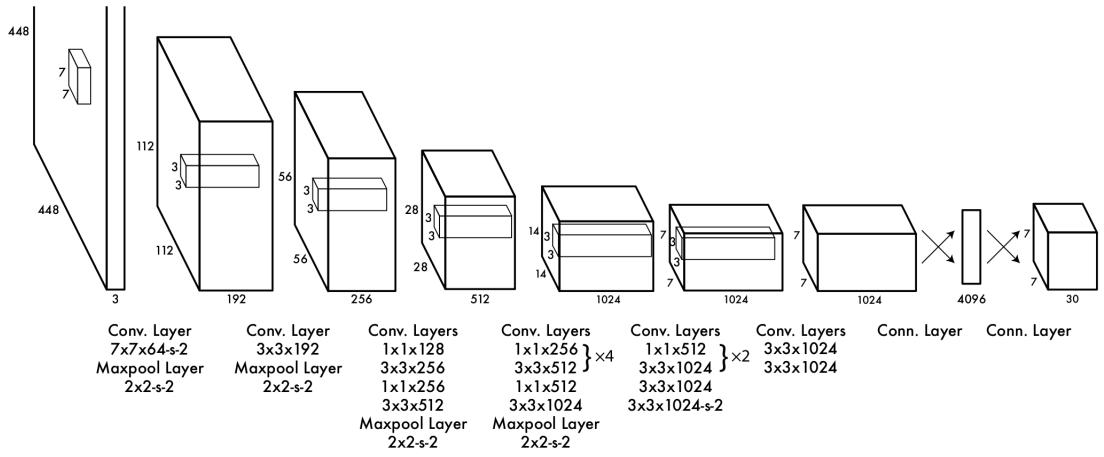



Figure 3.7: The one stage architecture of YOLO. Figure is taken from [Redmon et al., 2016].

This methods first generates bounding boxes for potential predictions, and then a classifier is run on those proposed predicted boxes to extract each required class. Basically, it first finds where there are possible objects you want to detect and then what are the objects that it found, in a give threshold score.

Because YOLO takes the entire image for training, the networks learns contextual information about classes together with their appearance in images as a whole, and consequently it is easily more generalizable for multiple types of classes (objects, animals, humans, etc.).

The model predicts a lot of bounding boxes for the classes it wants to detect, due to its architecture. This is good because it can handle limitations like occlusion (glasses on a face or rotated faces) decently well, but we then need to make use of thresholds to select the optimal results we want and also a non-maximum suppression function to remove overlapping bounding boxes and select the best one.

Other limitations are:

- Because of the bounding boxes prediction technique, this method struggles with objects that have very unusual shapes or aspect ratios.
- For the same reason, small overlapping objects that appear in groups are hard for this system to detect, and the use of non-maximum suppression function and Thresholding can significantly lower precision in those cases.
- This method is a little delicate in the training phase, as small errors in the training phase, can have effect on the intersection over union accuracy of the model.

For this method, the Darknet framework is used in training and inference. Darknet is an open source neural network framework used mainly for object detection.

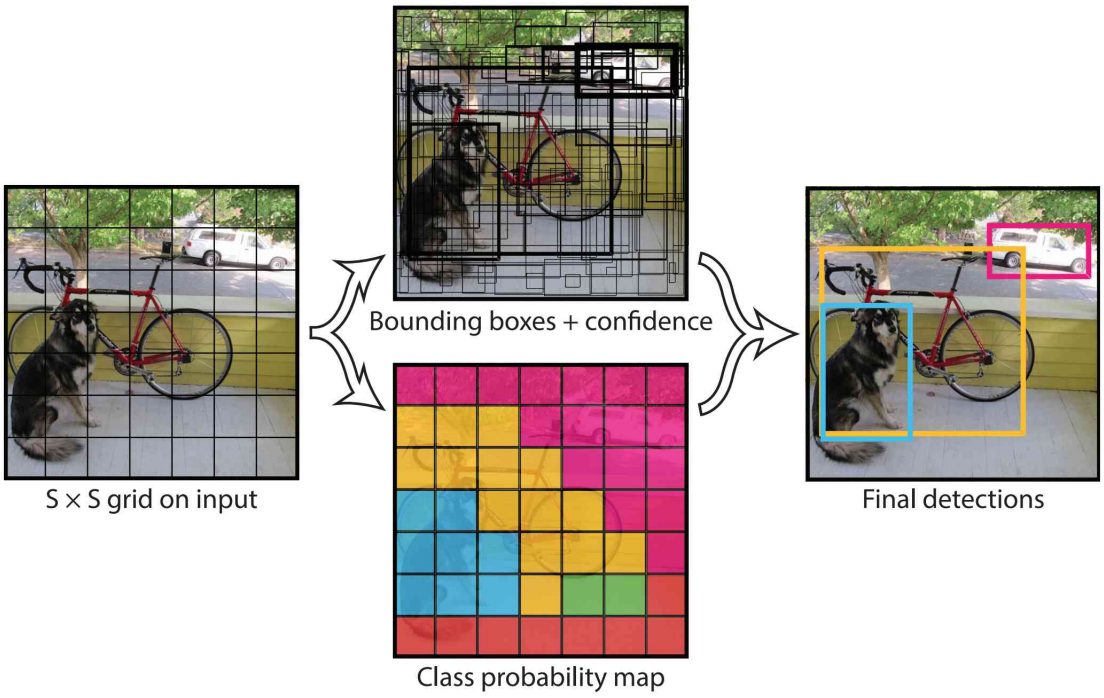



Figure 3.8: This figure displays a simple representation of YOLO, which acts as a regression problem. Figure is taken from [Shen, 2019].

I will not go more in depth about each layer and formula behind the first YOLO version. More information about how it was studied and invented can be found in the original paper for YOLOv1 [Redmon et al., 2016] There have been three versions of YOLO proposed by Joseph Redmon (YOLOv1, YOLOv2/YOLO9000 and YOLOv3).

The second version of YOLO was released in 2017 and had numerous improvements over the first version, in that it was faster and more accurate with an increased precision [Redmon and Farhadi, 2017]. The authors also proposed YOLO9000, a method to predict object classes that do not have annotated training data, by using a WordTree hierarchy of visual concepts.

One year later in 2018, the another official version of YOLO by Joseph Redmon had been released, YOLOv3 [Redmon and Farhadi, 2018]. Other third party variations of the YOLO algorithms were released after this, like YOLOv4 and YOLOv5, that improved certain aspects of YOLOv3, but they were by other authors.

YOLOv3 was the last version by the original author, which made various small improvements on the YOLOv2 system, but "nothing like super interesting", as stated by the authors. They used a new hybrid approach for feature extraction, using Darknet-19 and concepts from ResNet to train a new classifier network.

The reason for which the original researcher stopped the development of the YOLO algorithm is also very noteworthy. The authors wrote in the paper that:

"computer vision is already being put to questionable use, and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much." ([Redmon and Farhadi, 2018]).

They already seemed to believe that their research can be used in harmful ways, and Joseph later confirmed on Twitter that he stopped his Computer Vision research due to ethical concerns:

"because I saw the impact my work was having. I loved the work, but the military applications and privacy concerns eventually became impossible to ignore." ([yol, 2022c]).

This stirred a whole discussion about doing research and risking it to be used in unethically. I will come back to this topic in the Conclusion.

YOLOv3 is the algorithm used for this method implementation in my project. I chose it because it was the most performant out of all the official versions. The weights used by me were trained by [yol, 2022b] and [Gurkan et al., 2019] and they both performed fairly similar.

From my initial test, I immediately noticed the real time capabilities of this algorithm, together with the precision it had for detecting even smaller, most probably because it is trained on images of size 416×416. The image size is a trade-off between speed and minimum detection size. Occlusion and face rotation are handled very well, with the detector being able to detect faces only from half a face, faces covered by face masks, hair, or sunglasses.

I also had to set a suitable threshold that each prediction should pass to be considered a detection and apply a non-maximum suppression function based on threshold scores to get rid of overlapping bounding boxes.

Overall, this approach is very innovative and has had a big impact on object detection systems in recent years.

### Other Methods and state-of-the-art

As you might have notice, there are a lot of methods, some very different from others. The current state of the art in object detection can be split into two different categories:

1. One-stage methods (YOLO, SSD, RetinaNet)
2. Two-stage methods (Faster R-CNN, Mask R-CNN, Cascade R-CNN)

The first methods focus more on inference speed, but still keep a very good precision. And the second methods focus more on accuracy, but are being developed to be faster.

Generally, complex combinations of good parts from previous methods are developed and surpass state-of-the-art performances. But, new methods are still being discovered, just like DETR (in 2020), so we should still be on the lookout for other innovative approaches.

### Overlapping Symmetric Crops Algorithm

I have also separately created an algorithm that would improve the depth of the models, allowing for detecting smaller faces from input images or videos.

This algorithm equally splits the image into 4 overlapping crops, from a centre point in the initial image. This can be done to whatever wanted depth, so for each level, the 4 resulting crops will then be cropped into 4 more crops each.

For the faces right between crops to not be skipped, these crops need to be overlapping each other. Then, all the crops plus the initial big image are used in the detection method. This makes it so that small faces in the image will have the possibility to be detected in the smaller crops created.

Finally, all the images are resized to a standard size and put into a batch for the detection.

The complexity of this algorithm is:

$$Number\ of\ crops = \sum_{n=0}^{\infty} 4^{n}$$

where n is the depth to which you want the image to be split. So, with a bigger depth, there will be exponentially more images for the models to detect faces in, thus slowing down the speed of each method.

A major limitation of this algorithm is that images of very low resolutions, when cropped, will present problems in the detection and possibly increase the number of False Positives.

By applying this, there is also the problem of multiple bounding boxes on the same face. So, a Non-maximum Suppression function is applied to those bounding boxes to filter the best prediction, with the highest score, out of all of them, for each face.

This algorithm made the detection of small faces, that most approaches skipped, possible.

## 3.2 Facial Recognition

Facial Recognition has similar ways to Face Detection that it can be approached with, as it can also be a classification task.

Just like I have said in the Background section, 5 major different method types can be used. I have chosen the most significant ones to try: Holistic, Feature based and a Deep Learning method.

The first three methods have been implemented using the OpenCV framework, and the last method, the Deep Learning one, uses a model that was trained by the creator of DLib, and that I implemented using both OpenCV and DLib.

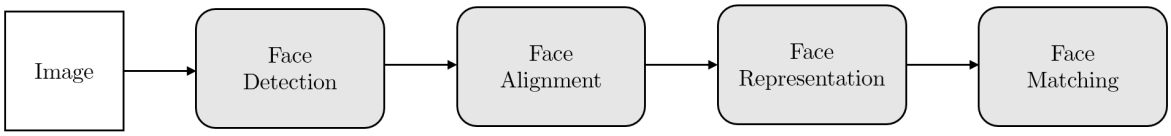



Figure 3.9: Those are the main steps for Face Recognition in images or videos that each method follows. Figure is taken from [Trigueros et al., 2018].

### 3.2.1 Eigenfaces

This method can be considered a **Holistic approach** that focuses on identifying variance between different parts of the face and features. Basically, it puts accent on the areas with the maximum change in intensity of pixels.

To make this possible, the **Principal component analysis** (PCA) process is used. It computes the principal components that best describes sets of data into unit vectors, that point to the directions and orientations of the data with the greatest variance. Those unit vectors, also called eigen vectors, are projected into a low dimensionality subspace, subsequently forming a covariance matrix.

The result from this computation are smaller sets of uncorrelated variables that highlight the variance of the input image/images containing a face. Therefore, the training images for this model must be already bounding as much of the face as possible and containing as little background as possible.

To make sure these methods performs optimally, a number of steps of pre-processing need to be addressed.

First, the bounding box of a face must be well-defined and catch at least from the forehead to the chin and each side of the cheeks.

Then, we need to make sure that the image is rotated to a proper standard position, otherwise the recognizer will struggle with the classification.

Next, the image must be a gray scale and of the same size as all the other images that are going to be used for training and testing.

This method is very affected by shadows and strong lighting, so to mitigate the negative effects from this, normalization of the face should be introduced.

Figure 3.10: Five computed Eigenfaces, sorted from the most variance (left) to the least variance (right). Figure is taken from [Trigueros et al., 2018].

Finally, as the method is a Holistic approach, improvements can be made by augmenting each class label of the model with different variations of the face. They can range from a flipped face to different contrasts and brightness, to be able to predict the true class in as many scenarios as possible.

The implementation of this method using OpenCV was not very complicated, but from initial tests and trials the results were not satisfactory as the effect of illumination and noise was impactful on False Positives.

### 3.2.2 Fisherfaces

This method is somewhat based on the Eigenfaces approach, but with added improvements to diminish the limitations.

Fisherfaces uses **Linear Discriminant Analysis** (LDA) instead of PCA. The PCA computes a linear combination formed by the features of the face, which maximizes the variance in the data, in a low dimensionality subspace, but it does not consider any classes in doing so, and thus discriminant information is lost in the reduction.

So, external factors like noise or excessive lighting can have a negative impact on the classification eigen vectors of the model.

As a result, LDA is introduced to try to alleviate this issue. LDA performs a dimensionality reduction that is **class specific** for each label in the training of the model.

To do this, a combination of features is selected from each class that best differentiates each other, so that variance in each class is reduced and variance between classes is maximized.

Simply put, similar classes are clustered together, and divergent classes are each furthered away in the low dimensionality representation of this method.

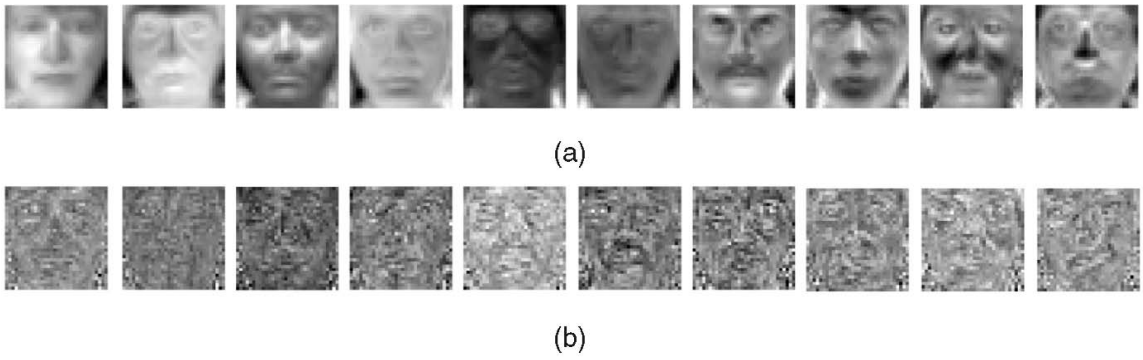



Figure 3.11: Visual representations calculated from face images in the YALE database using (a) Eigenfaces and (b) Fisherfaces feature vectors. It can be seen how Eigenfaces is significantly impacted by illumination and shadows and Fisherfaces is less accurate on the representation but also less impacted by external factors. Figure taken from [He et al., 2005].

This approach was meant to solve external factors, like illumination, impact on recognition, and while it was an improvement, it still has its limitations.

If the model learns faces with a type of camera, at a certain resolution, noise and size, and in certain well lit or badly illuminated scenarios, it will then struggle to recognize components from faces in different set-ups, other than the ones it was trained on.

In isolated experiments this model performs decently well, but in real life scenarios there would be too many issues.

And also, those two methods are very dependent on the input images, which is not perfect, but is logical because of the Holistic approach taken.

Initial experiments with this method provided visually little improvements to the previous method, and the results were still not adequate for real time and real life use in different scale invariant scenes.

### 3.2.3 Local Binary Patterns Histograms (LBPH)

**LBPH** can be considered a Hybrid approach, that also makes use of local features for performing the recognition. Local Binary Patterns are used to summarize the image into local regions by comparing each pixel with its neighbourhood.

A local structure is created this way, and then for each one a histogram is extracted, hence the name Local Binary Patterns Histograms.

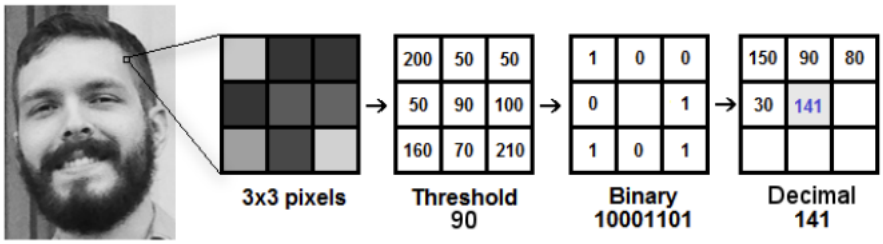



Figure 3.12: Creation of Local Binary Patterns of visual descriptors from an image. Figure taken from [Suma and Raga, 2018].

By making use of local features instead of vectors in a low dimensionality space, we focus on the key points of the faces that should be able to recognize them independently of what is going in the background or what external factors are present.

With the edges extracted from the training image, it is then theoretically easier to compare with the edges extracted from an evaluation image. Descriptors are encoded, and their histograms are checked with histograms from each class, thus making a prediction with a certain score on how close it matches to any of the other labels.

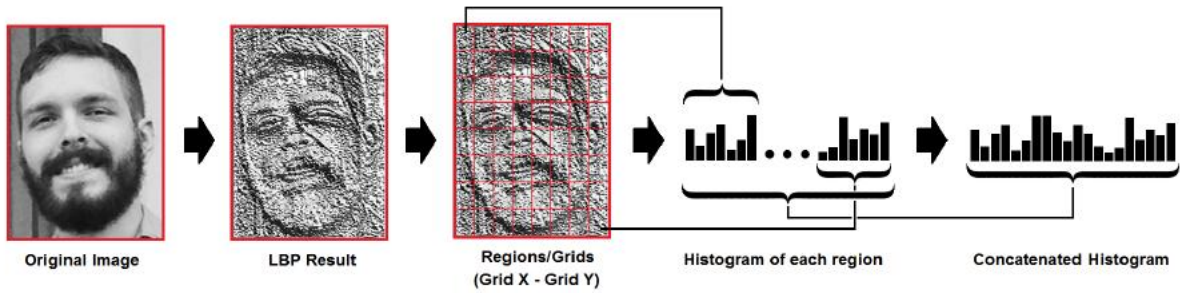



Figure 3.13: Computation of LBP Histogram that will be used for classification. Figure taken from [Suma and Raga, 2018].

This method contains concepts we have seen before in object detection, like HOG, but the task is approached a little differently, using extracted features to perform the classification.

LBPH was indeed noticeably better than the previous two methods, in the initial tests I ran, and while it still showed signs of performance loss in different set-ups and with different external factors, it remained fairly consistent in its precision and accuracy.

I would also like to note that for all the 3 non-Deep Learning approaches explored, I also tried various pre-processing steps to see if I can improve any one of them. Some of those steps were inspired from [Tan and Triggs, 2010], [Kazemi and Sullivan, 2014], and [geo, 2022].

DLib's shape predictor [King, 2009] was used to detect the most important landmarks of a face, and with those key points, I rotated the face appropriately so that every inputted face would have the same standard position.

The landmarks of the face are usually points that define the jaw, eyes, mouth, nose and eyebrows, so by aligning the eyes and mouth to a vertical position the face would be standing straight.

I implemented the Tan and Triggs pre-processing step as well. This algorithm tries to counter the effect illumination, shadowing, and noise can have on the feature extraction in an image.

This work by filtering and normalizing the input image before it goes into the feature extraction phase, so that each landmark is clearly visible for any outside factor effects.

You can read more about the algorithm and how it works in detail here [Tan and Triggs, 2010], and also see another example of its implementation here [geo, 2022].

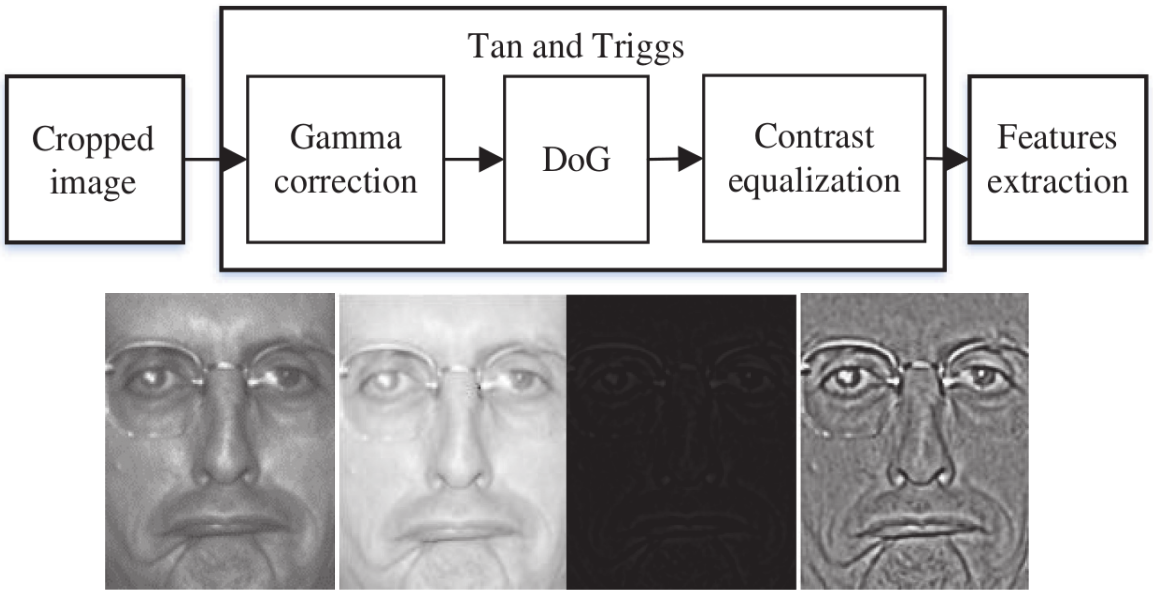



Figure 3.14: Steps of Tan and Triggs Pre-processing and image examples. The first image is gray scaled, second has Gamma correction applied, third has Difference of Gaussians (DoG) filtering and fourth has Contrast equalization and is the output image. Figure is taken from [Ghorbel et al., 2020].

As it can be noticed in the figure above in the last image, the edges are more prominent and thus make the feature extraction more precise.

I decided on adding extra pre-processing steps to learn more about the Image Processing field and results it can achieve.

So, I included a function for histogram equalization and a function to create an elliptical mask around the face.

The histogram equalization is meant to normalize the consequences lighting, brightness and contrast can have in the LBP Histogram creation step.

An elliptical mask can be used to hide the background of a square image of a face, thus reducing unnecessary information from the background.

The background is blacked out, starting from the edges, up to a certain size, and so the face is enclosed in an ellipse. But this step presents a problem because it will discriminate faces that have different aspect ratios, for which this mask will sometimes cover important parts of the face.

For this reason, I decided that this step can end up doing more harm than good, and did not use it.

Overall, those 3 methods were good for isolated experiments and tests, but it was clear that for a real life systems, like a security camera, they were still lacking.

Even with all the added pre-processing, I still had to look for improvements in Facial Recognition methods, therefore I decided on exploring a Deep Learning approach.

### 3.2.4 DLib ResNet Neural Network Face Recognition model

This method was presented by Davis King in the DLib Machine Learning with Computer Vision framework [King, 2009]. It uses a version of a Residual Neural Network, ResNet-34, a Deep Learning approach to Facial Recognition.

This model is composed of 29 convolutional layers, trained from scratch on 3 million faces from the VGG dataset [vgg, 2022] and face scrub dataset [fac, 2022a], with 7485 individual identities.

The initial weights for training this model were randomly defined, and a hinge loss function was used to ensure the correct learning of the classifier to identify each facial feature for each different person's face.

The way this model works for new input faces it that first you feed an image of a face into the network and get back descriptor vectors for this particular face class.

Those descriptors are believed to be based on the whole face, with accent put on landmarks and key points for each class. But we can not tell for sure what exactly the Neural Networks computes as it uses a Deep Learning approach.

You also need to use a shape model predictor to align the face straight in the image, for the network to be able to detect the features correctly.

After you have the vectors of face descriptors for the label classes you want to identify saved, new images can be inputted for evaluation into the network for evaluation.

Those new face descriptors received for evaluation are then compared to all the current saved face descriptors that have been labelled.

The Euclidean Distance is computed for each label, and the one with the lowest distance is chosen. Then, if the one with the lowest distance is under a set threshold, we can consider the new face to be identified to the descriptor label that it matched to.

From this we notice that the more face descriptor classes we have, the slower the face identification will be, and the higher the chance of a False Positive. With that said, it was the most robust method I tried, out of all 4, and still performed with real time speed in my experiments.

The biggest advantage of this method is that Deep Learning is used to classify which face descriptor vectors are best used for human recognition.

And so, with the Machine Learning techniques available for optimization of Neural Networks, like the loss function, plus the exhaustive training and learning that the ResNet classifier network can comprehend, this model provides a decently precise and accurate facial recognizer.

Even with effects from harsh external factors and limitations, such as illumination, occlusion or noise, the results of this method are good.

#### Other Methods and state-of-the-art

DNNs, more specifically CNNs have been achieving top state-of-the-art performance in facial recognition from images [Balaban, 2015].

As I have mentioned before, apart from those 5 classic methods for recognizing faces in images, there are also more advanced approaches that require more data input. Depth sensors are very good at facial identification, using various measures in recognition. For example, Apple's FaceID "sees" the face in 3D and uses 30000 invisible infrared dots to properly capture and recognize it.

Overall, developments have made it far enough for us to be able to use face identification biometrics in everyday devices, like smartphones, thus reaching a satisfactory confidence level. But I believe there is always room for improvement.

## 3.3 Emotion Detection

For this task, there can be a few approaches possible.

One way is to extract the landmarks of a face and geometrically compute them, possibly with the help of machine learning, to get approximations of different expressions that the face can show.

So, a smile, sad mouth or raised eyebrows can be identified, but this would certainly not be enough to catch a complex number of emotions, apart from simple expressions.

A much better approach to recognizing emotions in images or videos would be classification using Deep Learning, either supervised or unsupervised.

I decided on doing this approach by using the Tensorflow open source library for Machine Learning and Deep Learning [Abadi et al., 2015].

The major reasons for utilizing TensorFlow were its capabilities for handling Deep Learning models and the potential it showed for integrating those respective models into any program, application or web application, through a lot of possible programming languages or any preferred methods.

Even though my main code was written in C++, I did this part using Python and Jupyter, as they were more appropriate in this use case.

And because of the framework used and the OpenCV DNN module, I was able to fully integrate the models into my program, relatively with ease.

What I did was I had selected some classifying pre-trained models from the TensorFlow Hub [Abadi et al., 2015] and then reused it as a starting point for a model for my needed task.

This action is called Transfer Learning, where I apply a solution from one problem to another similar problem I want to solve.

I then performed Fine-tuning of those models to adjust them to better fit my intended purpose, detecting and classifying emotions.

The database I have used for the learning of my models is called Real-world Affective Faces (RAF) Database [Li et al., 2017], provided to me by Shan Li, from whom I requested access to.

It contains 15339 images of people, each labelled with 1 out of 7 expressions: Surprise, Fear, Disgust, Happiness, Sadness, Anger, Neutral.

This allowed me to fine tune models for detecting those 7 basic emotions in a face, but due to imbalance in the number of images for each class, with happiness and neutral emotions being the most prominent, my model is also more prone to classify low confidence predictions with those emotions.

The models I experimented with were both pre-trained on the ImageNet [Deng et al., 2009] data set. The ImageNet data set contains 14197122 images and thousands of classes for them. So, I predict those models will handle my smaller data set with 7 classes fairly well.

Those two models are:

### 3.3.1 EfficientNet

**EfficientNet** is a type of image classification model, based on a Convolutional Neural Network architecture [Tan and Le, 2019].

The purpose of this model was to make a small scale network, that can be integrated on less demanding hardware. While it is made to be fast, it still maintains a great accuracy with its classifications.

So, this model can be implemented on small and compact systems that still would benefit a lot from good precision and speed, like smartphone devices or digital cameras.

The specific model I have used is a EfficientNet-B0, a mobile-size baseline network, but more scaled up versions of EfficientNet are available from EfficientNet-B1 to EfficientNet-B7. The B0 version I used had roughly 4 million trainable parameters with 237 layers in total, with the layers being combined in 5 module types to ease computation.

This model was also proposed with Transfer Learning in mind, so that people can take the pre-trained weights and use them for different classification tasks, which is exactly what I did. This was proven by the researchers of EfficientNet in their initial release paper [Tan and Le, 2019].

Additionally, I have extended the EfficientNet model with a few more layers to make it possible to perform Transfer Learning.

I had to add an input layer, a global average pooling layer, an activation layer and a dense output layer.

The input layer handles the training images I gave it; the pooling layer reduces the dimensions of the feature maps, thus the number of training parameters is reduced and computation is faster; the activation layers dictates the training of the model and what is to be learned from all the input, using the Softmax "probability" function; and the dense layers is used to output the prediction of the classifier to a certain class.

### 3.3.2 Residual Neural Network (ResNet)

**ResNet** is also a type of image classification model, based on a CNN architecture, but with a more variable number of layers and parameters.

ResNets use a Deep Residual learning framework that allows for an easier training and optimization process, this whole process being proposed in the paper [He et al., 2016a] in 2015.

This is done by introducing skip connections over some layers of the network to improve the convergence of the model.

So, ResNets allow construction of much deeper Neural Networks, without having exploding or vanishing gradient problems, or degradation of accuracy.

The increased number of layers will then improve the classification functions results of the Neural Network, with the added benefits of not having the respective limitations and problems.

The particular model I decided on using was ResNetV2-50 [He et al., 2016b], developed one year later by the same researchers.

It is a CNN comprised of 48 convolutional layers, 1 MaxPool layer and 1 AveragePool layer, with a little over 23 million trainable parameters, much higher than the 4 million of EfficientNet-B0.

The major difference between V1 and V2 is that V2 uses a batch normalization technique on the input before each weight layer to increase the learning rate of the model and reduce the training time of the network through the batches it is served.

This certainly helps in the Transfer Learning and Fine-tuning of this model for Emotion Detection.

With that said, I still had to add 2 more necessary layers, an activation layer using Softmax and a dense output layer, just like I did in the EfficientNet model.

**Implementation**

After the fine-tuning of the model, I optimized it by removing some redundant layers that were no longer needed for inference.

Finally, I had to freeze the graph of the model, fusing its layers together, to be able to optimally integrate it into my program. TensorFlow fortunately had a few function that made this task manageable.

With all that said, the biggest limitation in Deep Learning is getting enough data for your model.

This was also the case with for me, while I had 15339 images and the people in them were of various ages and looks, the amount of training data proved to be insufficient.

My experiments showed that glasses, especially sunglasses, plus occlusion and orientation of the face have a very big effect on the prediction.

I have tried my best to diminish those effects by augmenting the data. In a random and with a fixed limit I have applied:

- Random image flipping
- Random image resizing
- Brightness augmentation
- Contrast augmentation
- Saturation augmentation
- Hue augmentation

Those proved a small improvement, but nothing major, as the data set would still be a little small for detection of 7 emotions in the wild.

With this I managed to implement a pre-trained model, with my own Fine-tuned weights, for Emotion Detection from the 7 emotions in the used database. Then my program could forward to the model in real time the bounding boxes from my face detectors and the model could classify emotions based on those.

### Other Methods and state-of-the-art

Emotion detection from images is basically a classification task, and at the moment the current state of the art for Image Classification with Deep Learning are considered to be CNNs.

With that said, more complexity can be added to improve facial expression classification approaches.

A good example would be adding two extra models that would perform some extra pre-processing steps.

Different aged people can have a lot of variety in how their face looks. To lower this effect, a gender, and an age model can be used to first predict some details about a person's face, and then the face plus this data can be forwarded to the emotion detector.

Because people have face so different from one another, implementations like this might be able to narrow down the information as much as they can, in order to help the classifier get a more accurate prediction.

Furthermore, compound expressions could possibly be used to make a Neural Network better understand emotions. Compound emotions would be formed of 2 or more emotions, just as in real life, when we can have multiple expressions showing on our face.

We can be surprised and happy, fearful and disgusted, sad and angry, etc. And face expressions are not exactly an on or off action, some people might look happier than others, so maybe a percentage label metric for the data can positively impact the model.

Apart from augmenting the current available data, a more complicated solution to the data set limitation of Deep Learning would be to artificially generate data samples.

Generative Adversarial Networks (GANs), an image and video generation model architecture, has the potential of being used to do this. It could create synthetic instances of people with various facial expressions.

But still, for improvements to be made, we would need to make sure that the generated data is almost perfectly resembling real life faces with appropriate facial expressions.

A lot of methods are available for this classification tasks, and many more are being developed.

# Chapter 4

# Results

## 4.1 Application and Program

I have implemented all the approaches listed above into a C++ code base to test and experiment with.

As C++ is very close to the machine language and is compiled into assembly code for the system, the main focus can be put on evaluating each method for Face Detection, Face Recognition and Emotion Recognition.

**Integration possibilities:** Furthermore, the choice of the language enables a better integration of the models into multiple types of devices, systems or applications, while allowing an easier porting of the code and algorithms into any other language.

**Reason and use:** I have written my code with modularity in mind. Each method works independently of one another, thus allowing for separate performance improvements and experiments, plus an easier evaluation.

To demonstrate the integration capabilities of those ideas, I created an "API-like" program that would run any methods with any variations on sets of images or videos. Using a bash script I could get coordinates of face bounding boxes, plus recognized people and emotions in a .csv file from any input data.

My program could also output sequences of images and videos with the bounding boxes around faces, the recognized faces names and the emotions, similar to an API.

This significantly helped in performing the comparison of my approaches and discovering any errors or problems in the implementation.

With those programs and scripts, I was able to perform a thorough analysis on my methods.

## 4.2 Evaluation and Comparison

### 4.2.1 Face Detection

For the evaluation of the 6 methods I implemented, I used the program to get bounding boxes predictions on a set of images from WIDER FACE testing and evaluation data [Yang et al., 2016].

I used 1694 images that had, in total, 10879 bounding boxes of faces.

This data set benefited from true labelled bounding boxes annotations for every face in every image, that I could use to compare each method from one another.

The 6 methods are: Viola-Jones = haar; HOG + SVM = hog; MMOD CNN = mmod; DNN-SSD Caffe = caffe; OpenCV DNN Tensorflow = tensor; YOLOv3 = yolo.

To be noted is that this Face Detection benchmark is a very complicated set of data because it contains a lot of small, occluded, noisy and blurred faces.

I have used 4 metrics to assert the performance of the approaches:

- Precision
- Recall
- F1 Score Accuracy (harmonic mean of precision and recall)
- Intersection Over Union Score

With those metrics I can determine how many detected faces are correct (Precision), how many faces out of all of them each approach detected (Recall), the overall balanced score (F1 Score) and how good does the bounding box fit the actual face (Intersection Over Union).

For this I had to retrieve the True Positive detections, how many bounding boxes have an IOU Score over a certain threshold; for my experiment I have used 0.5.

This means that a predicted bounding box intersecting over 50% with a true bounding box will be counted as a TP.

The False Negatives are all the faces that have not been detected in an image.

And the False Positives are all the predicted bounding boxes that do not intersect with any true bounding box over a face.

The raw results are:

| Total 10879 bounding boxes | Haar | HOG | MMOD | Caffe | Tensor | YOLO |
|---|---|---|---|---|---|---|
| True Positives | 2903 | 978 | 4182 | 4676 | 4102 | 7451 |
| False Positives | 741 | 41 | 220 | 289 | 217 | 1025 |
| False Negatives | 7976 | 9901 | 6697 | 6203 | 6777 | 3428 |

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{4.1}$$

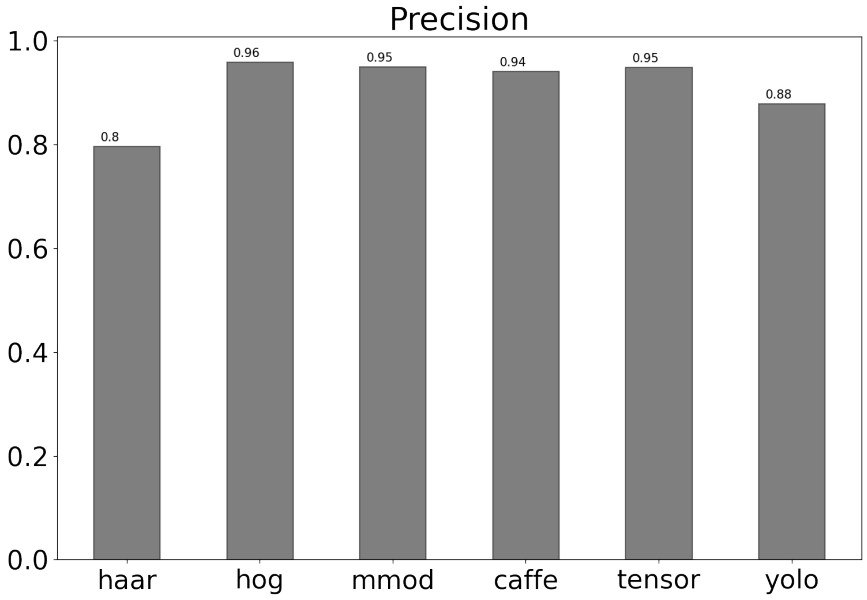



Figure 4.1: Precision bar plot for the 6 methods.

In the Precision metric we can notice that SSD Caffe, SSD Tensor, HOG and MMOD perform very well at around 95% each, with YOLO and Haar lacking a bit behind.

No major difference can be pointed out in False Positive detections between Classic methods and Deep Learning methods.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.2}$$

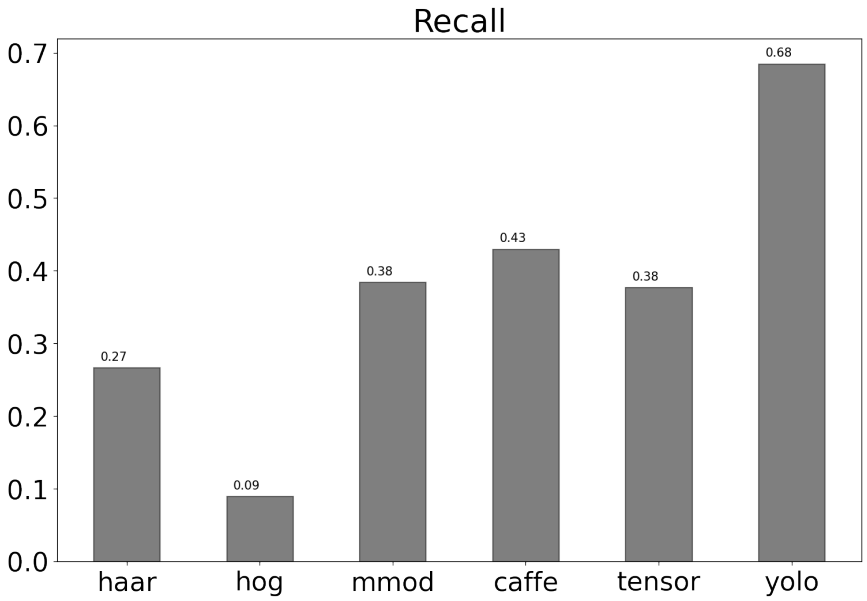



Figure 4.2: Recall bar plot for the 6 methods.

The Recall metric is very interesting because we notice the YOLO method outperforming all the other methods, by detecting the most faces in the data images, at 68%.

The Deep Learning methods Caffe, Tensor and MMOD have still a fairly good Recall, considering that the data set used contains a lot of blurred and small faces.

Viola-Jones Haar is a bit behind them and HOG has the lowest metric.

$$F1\ Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4.3}$$

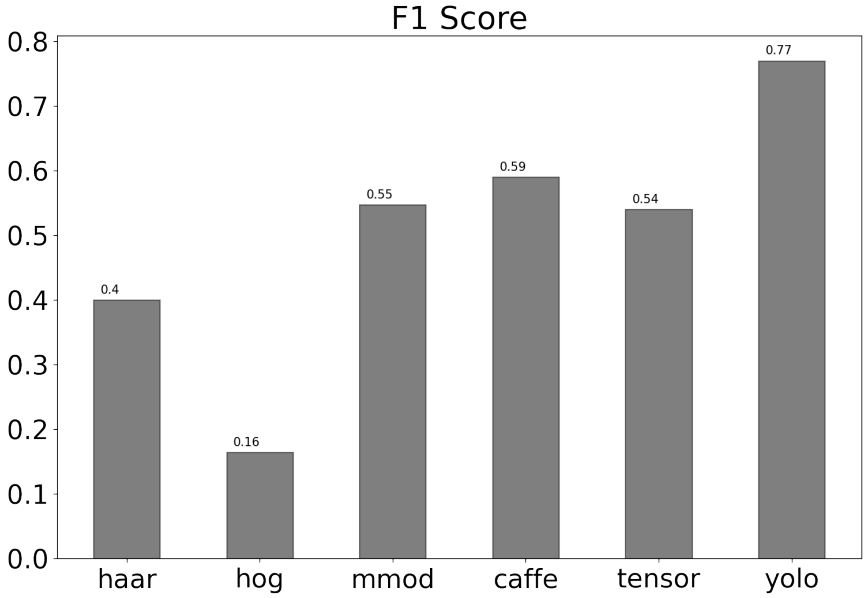



Figure 4.3: F1 Score bar plot for the 6 methods.

In the overall F1 Score we can see the that YOLO is closer to MMOD, Caffe, Tensor methods, but there is still a big difference between them and Haar, and especially HOG.

$$IOU = \frac{(BBox\ A) \cap (BBox\ B)}{(BBox\ A) \cup (BBox\ B)} \tag{4.4}$$

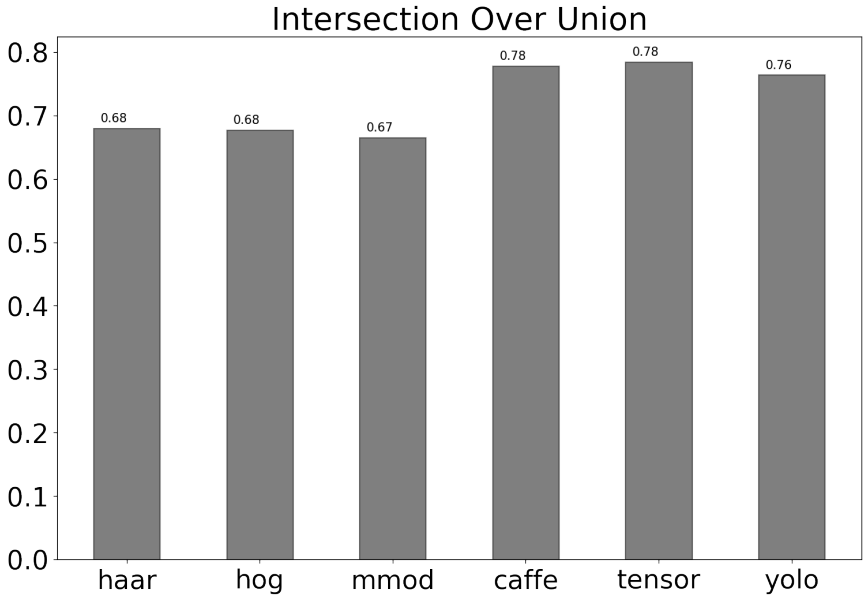



Figure 4.4: IOU Score bar plot for the 6 methods.

I consider this metric fairly important because it dictates how well we can perform Face Recognition and Emotion Recognition.

We need a good IOU score so that face features can be extracted to perform classification on them.

Caffe, Tensor and YOLO perform well, and Haar HOG and MMOD are still good but with a lower IOU Score.

| Results for 720×720 resolution | Haar | HOG | MMOD | Caffe | Tensor | YOLO |
|---|---|---|---|---|---|---|
| FPS on Real-time videos | 55 | 2 | 45 | 80-85 | 85-90 | 40 |
| Seconds it takes to process a frame | ∼0.02 | ∼0.5 | ∼0.02 | ∼0.01 | ∼0.01 | ∼0.025 |

Table 4.1: Approximation of processing time performance for methods comparison

The speed performance of the methods is important to consider when comparing the methods.

As expected, the SSD One Stage Detectors are faster than Haar and HOG, and also have a much better accuracy performance.

The HOG method is slower because of the image conversions I had to implement from OpenCV to DLib and back, so the metric is not the most accurate, with the method possibly being able to run a bit faster.

The YOLO method is not theoretically faster than the other methods, but has the added detection of very small faces, because the model was trained to look for that small object or faces.

This is the reason it has the highest Recall score out of all the approaches, but it is slower than the other One Stage Detectors.

This highlights the trade-off between Precision, Recall, IOU, and Speed.

Most of the time we can create Deep Learning models to be faster but look less deep into the input, or do the other way around if we need more accurate results.

The OpenCV Tensor method uses the same framework and architecture as the Caffe method, but is quantized using TensorFlow to reduce the size of the model.

While the model is half the size of the Caffe one, we notice that the results are very close to each other. So this technique can be very useful when implementing this type of models in small devices that have less processing power and space.

I also want to show an evaluation of the methods without the Overlapping Symmetric Crops Algorithm I implemented.

YOLO is already trained to detect small faces, so the overall score does not change, but for the other 5 methods we notice a significant improvement in the performance when using the Overlapping Symmetric Crops Algorithm with a depth of 3.

This means that the image is split into:

$$4^0 + 4^1 + 4^2 + 4^3 = 85$$

crops for the detection of much smaller faces.

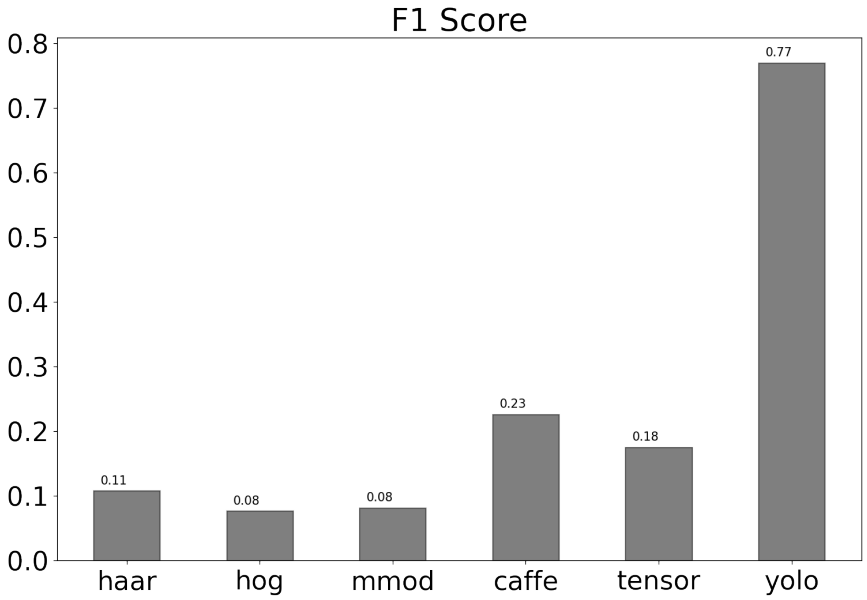



Figure 4.5: F1 Score for the 6 methods, without the Overlapping Symmetric Crops Algorithm.

The Deep Learning approaches outperformed the Haar and HOG by a wide margin.

It is true that the data set used was very harsh on the evaluation, with very hard to detect faces, but it best represent detecting faces “in the wild”.

### 4.2.2 Facial Recognition

I evaluated the 4 Facial Recognition methods I implemented using annotated faces images from the Labelled Faces in the Wild data set [Huang et al., 2007].

I selected 4 people, and 1027 images of them to test out my approaches, and using my program and the Caffe Face Detector I tried to match them to the faces in the images.

I was not able to compare all 4 of them on a much bigger number of labels because Eigenfaces and Fisherfaces were extremely slow when handling hundreds of classes with thousands of images. LPBH was doing fairly well in terms of processing speed, but the predictions were visibly not good.

The Deep Learning approach could handle a few hundreds of classes with thousands of test images, with decent results.

I decided that testing them on this data would be enough to highlight their performance comparisons.

The results for the experiment are:

| 1027 total faces | Eigenfaces | Fisherfaces | LBPH | ResNet |
|---|---|---|---|---|
| Correct Predictions | 279 | 247 | 381 | 1009 |
| Wrong Predictions | 748 | 780 | 646 | 18 |
| Accuracy | 27% | 24% | 37% | 98% |

As I expected, the Deep Learning method performed the best, and at a good speed. LBPH has a good enough accuracy, but still not satisfactory.

The Eigenfaces and Fisherfaces are slower and have a much lower accuracy.

The faces in the data set are heavily impacted by external factors, and the people in them have various postures, so it is not easy for the Holistic approaches to recognize them.

Also, to remember is that a number of facial recognition methods are dependent on the camera resolution and noise they are trained and tested with. If they differ, the performance starts decreasing.

The best approaches for recognizing faces "in the wild" are the ones that extract only important features and key points from a face and compare them, while lowering the outside factors, that they can not control, impact.

### 4.2.3 Emotion Detection

I performed transfer learning on EfficientNet-B0 and ResNetv2-50, fine-tuning them to be able to classify the 7 facial expressions mentioned before.

With the help of the TensorFlow framework, I was able to split the data into 3 parts; training (80%), evaluation (10%) and testing (10%).

The data set used was presented in "Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild" [Li et al., 2017], and has, 15339 distinct faces with labelled expressions.

As both models used are already very performant and have state-of-the-art results, training went very good for both of them.

Below are graphs for accuracy and loss of both methods for the 200 epochs I ran the training:

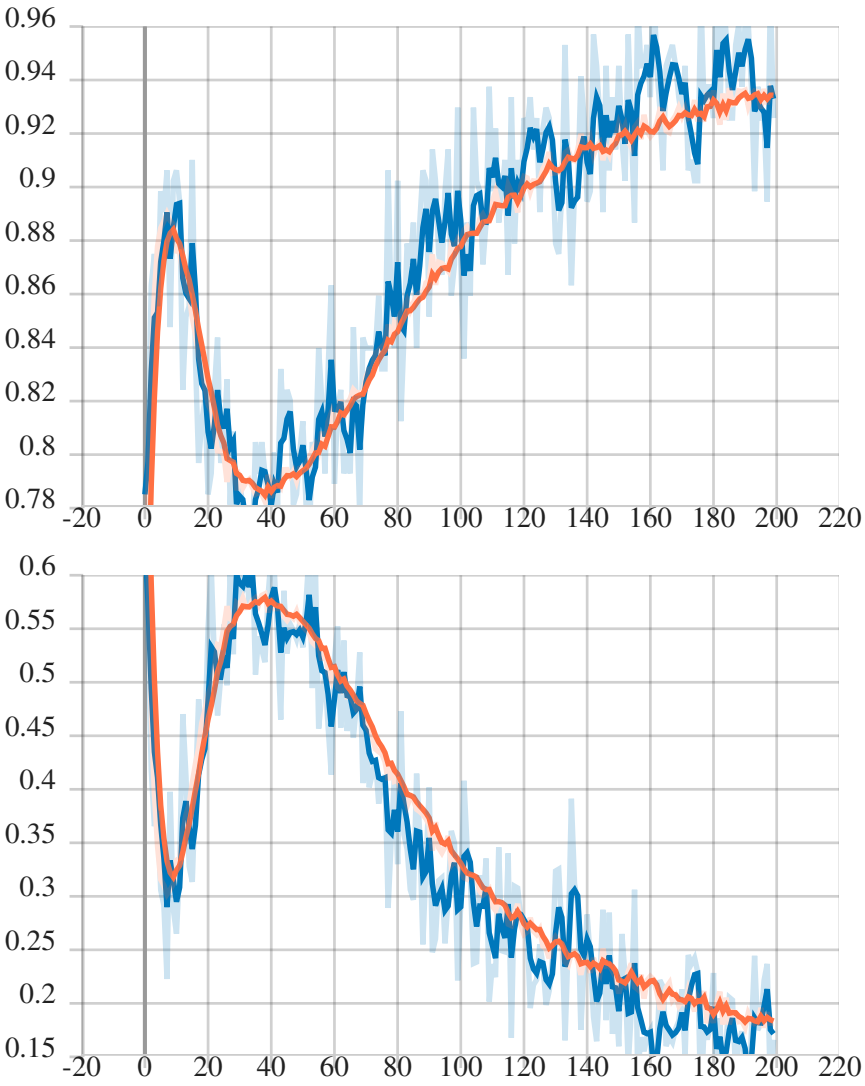



Figure 4.6: EfficientNet-B0 accuracy (upper) and loss (lower) by epochs; train values are orange and evaluation values are blue.

For EfficientNet, the training accuracy reached a good 94%, as did the evaluation accuracy.

The loss function also performed very well and the value goes towards 0, meaning that the model is learning.

The accuracy on the test data set, which was separated from the training data set, reached an accuracy of 94%, with a loss of 0.17.

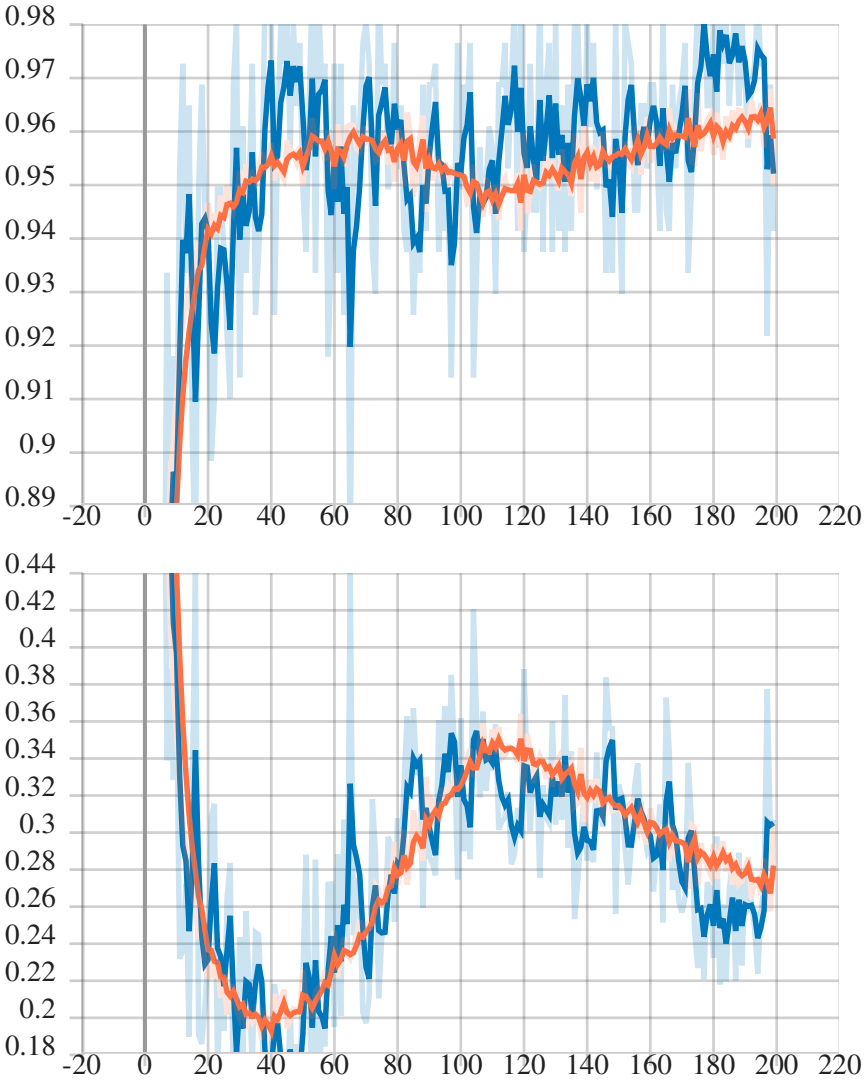



Figure 4.7: ResNetv2-50 accuracy (upper) and loss (lower) by epochs; train values are orange and evaluation values are blue.

For ResNet, the training accuracy also reached a very good 96%, and the evaluation accuracy fluctuated a bit more, but reached roughly in the same range of 95%-97%.

The loss metric is a little higher than the EfficientNet loss, possibly because of the significantly bigger parameters size of ResNet, but it is still going towards 0.

The ResNet model would need more time and data to train so that the raw prediction confidence would be higher for each prediction, thus making the loss smaller.

The testing accuracy of this method is 97%, with a loss of 0.23.

Both methods achieved good results, but each method has its advantages and disadvantages.

EfficientNet has less trainable parameters, thus making it easier and faster to train, but is also less deep and could possibly face some issues with much bigger data sets when trying to understand each emotion and differentiate them from each other.

ResNet on the other hand is slower to train and requires more processing power, but can recognize emotions more profoundly, by an appropriate confidence variable.

This model I believe can be trained for a longer amount of time with bigger data sets and get better results for expressions “in the wild”.

A problem that was also noted in the data set paper [Li et al., 2017] is that the number of classes for each expression is not equal.

- Happiness: 5957 occurrences
- Neutral: 3204 occurrences
- Sadness: 2460 occurrences
- Surprise: 1619 occurrences
- Disgust: 877 occurrences
- Anger: 867 occurrences
- Fear: 355 occurrences

These imbalances makes it so that the model will lean over to understanding and predicting more faces into the dominant classes, which is certainly not good and can be addressed by finding a more appropriate data set.

We can not rely only on the evaluation of the models on this data set in this training and testing environment.

This classification experiment is done on only images from a database, not on people in unconstrained situations and their facial expressions.

It is very hard and expensive to train and test a model for a task like this because the end goal would be to recognize expressions on people in any circumstance, with impacts from any outside factors.

So, occlusion, face rotation, illumination, shadowing, etc. can significantly decrease the accuracy of the predictions made in real life situations.

From more experiments with the model that I exported into my program, I have noticed that it detects expressions well only when the face in a fully frontal position to the camera and there are not a lot of occlusion problems, like sunglasses.

Overall, as it is always with Deep Learning, the biggest limitation of this method is what the models are trained on.

The performance relies heavily on the versatility and size of the data set, the more edge cases with external factors it has, the better the results will be.

Usually data can be augmented and manipulated to fit those needs, but with human faces it is considerably harder.

# Chapter 5

# Conclusion

This exploration and research for detecting and classifying humans and their features, provided me with great insight into the world of Computer Vision, Machine Learning and Deep Learning.

It has also highlighted the major developments that have been happening in the past decades, allowing machines to understand images and videos, and especially in the past few years, with the significant evolution of Neural Networks in Computer Science.

I have created a codebase, supported by a report and results, that would introduce anyone into the field of Computer Vision with Machine Learning and Deep Learning.

## 5.1 Results Discussion

It is well known that the introduction of Deep Learning advanced the performance of methods and systems in terms of speed and accuracy, and Computer Vision is not an exception.

From the Face Detection and Face Recognition results, I noticed that Deep Learning methods outperform most of the classic methods available in all areas.

They are also easier to implement and use in our code. Classic methods usually require a lot more "nit-picking" to get them to a satisfactory performance.

In the Emotion Recognition approach, I display how already made Neural Networks can be trained and perfected to handle the tasks we want them to; we only need to choose the correct frameworks and models.

Both ResNetv2-50 and EffieicnetNet-B0 have good results, but with more data and memory available, deeper Neural Networks, like EfficientNet-B7 ( 65 million parameters) and ResNetXt-101 ( 85 million parameters), can be trained on different classifications problems and provide better results.

Keep in mind that larger models will have increased speed performance decay.

Still, Deep Learning methods have a very big problem that one has to overcome to be able to train and use them, the available data sets that it requires.

Data sets dictate how well the Deep Learning method performs, so it is very important to find the right data set for a task and manipulate it to best assist the Network in its training.

With all that said, most of the time we have to decide on what approaches we want to use, based on their trade-offs.

Each method has its advantage and disadvantage, as it can be seen in the YOLO and Caffe/Tensor approaches. YOLO finds much smaller faces, with a little lower Precision and IOU Score and Caffe/Tensor do not look for faces that small, but have a better Precision and IOU Score, plus are faster than YOLO. Both of them are Single Stage Detectors, but they are trained differently.

And those are not the only differences, models can vary in size and processing power needs, which is something important to consider when implementing those approaches into smaller devices.

The best approaches are the ones that suit your needs, and that is why we have to know how to research about each method and compare it to other available method. This way, we can make sure that we can choose a right approach for the task at hand.

## 5.2 Ethical Consideration

As with most technologies, this research subject is not without its critics.

Face and Object Detection could possibly be used in harmful scenarios, and a lot of researchers do not want to be involved in these awful practices.

Others worry that Face Recognition could be used to invade people's privacy or to unfairly target certain groups of people.

And the research and evolution of this domain is influenced by this, as it was seen with the initial creator of the YOLO algorithm, who stopped his research due to the impact it had on some questionable practices by certain organizations.

While most of the time, researches are not directly responsible for those actions, I do believe that raising awareness to this matter is still very important, and every computer scientist should take ethics into consideration.

## 5.3 Reflection and Future Work

I am very positive and excited about the new possibilities that this kind of technologies have opened for us, in terms of digital evolution and research.

While I was not able to explore every possible method for detection and classification in images and go much in detail about how each presented approach mathematically works, I hope that everyone reading this will have gained at least a little interest in this technology.

And I wish this report will serve as an introduction to the opportunities and challenges put before us in Computer Vision and Deep Learning, and how each of us can go into this area and help develop it further.

There is still a lot of work to be done for this, and I will continue my research into this field that I find so interesting, while hoping other, maybe inspired by this report, will join me in making progress.

# Bibliography


[tra, 2022a] (Accessed 10.4.2022a). Face detection using transformers explanation https://becominghuman.ai/transformers-in-vision-e2e87b739feb.

[ima, 2022] (Accessed 10.4.2022). Image processing algorithms website https://www.imagproc.com.

[med, 2022c] (Accessed 12.4.2022c). Integration of tensorflow graphs in c/c++ https://medium.com/jim-fleming/loading-a-tensorflow-graph-with-the-c-api-4caaff88463f#.krslipabt.

[tf, 2022] (Accessed 12.4.2022). Tensorflow page https://www.tensorflow.org/.

[blo, 2022] (Accessed 12.4.2022). Training your own dnn and exporting it project http://chembl.blogspot.com/2020/04/mini-project-training-nn-network-in.html.

[tra, 2022b] (Accessed 12.4.2022b). Transfer learning and fine tuning helping material https://towardsdatascience.com/image-classification-transfer-learning-and-fine-tuning-using-tensorflow-a791baf9dbf3.

[tfb, 2022] (Accessed 12.4.2022). Transfer learning blog https://blog.tensorflow.org/2020/05/bigtransfer-bit-state-of-art-transfer-learning-computer-vision.html.

[yol, 2022c] (Accessed 13.4.2022c). Tweet of yolo creator regarding ethical issues https://twitter.com/pjreddie/status/1230524770350817280?

[dli, 2022] (Accessed 1.4.2022). Dlib homepage http://dlib.net/.

[geo, 2022] (Accessed 1.4.2022). Github page with a face recognition method implementation https://github.com/GeorgeSeif/Face-Recognition.

[lea, 2022] (Accessed 1.4.2022). Learnopencv homepage https://github.com/spmallick/learnopencv.

[ope, 2022] (Accessed 1.4.2022). Opencv homepages https://opencv.org/ and https://docs.opencv.org/.

[vgg, 2022] (Accessed 14.4.2022). Dataset publication website https://www.robots.ox.ac.uk/~vgg/data/vgg_face/.

[fac, 2022a] (Accessed 14.4.2022a). Dataset publication website http://vintage.winklerbros.net/facescrub.html.

[sam, 2022a] (Accessed 21 April 2022a). Copyright free images from https://pixabay.com/, https://www.pexels.com/, https://unsplash.com/.

[sam, 2022b] (Accessed 21 April 2022b). Photo by carolyn kaster in associated press posted in the daily mail https://www.dailymail.co.uk/news/article-3581488/33-years-later-NC-State-championship-team-meets-president.html.

[sam, 2022c] (Accessed 21 April 2022c). https://www.languageoftheface.com/emotiongallery2.html.

[sam, 2022d] (Accessed 21 April 2022d). https://www.scienceofpeople.com/microexpressions/.

[yol, 2022a] (Accessed 2.4.2022a). Github page of yolo implementation https://github.com/msprITU/YOLOv3-Face.

[yol, 2022b] (Accessed 2.4.2022b). Github page of yolo implementation https://github.com/sthanhng/yoloface.

[mlm, 2022] (Accessed 3.4.2022). Face detection guide and tutorial https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/.

[pyi, 2022] (Accessed 3.4.2022). Face detection in opencv using deep learning https://pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/.

[med, 2022a] (Accessed 3.4.2022a). Face detection state of the art methods https://medium.com/sciforce/face-detection-explained-state-of-the-art-methods-and-best-tools-f730fca16294.

[med, 2022d] (Accessed 3.4.2022d). Mtcnn method for face detection https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923.

[fac, 2022b] (Accessed 4.4.2022b). Deep learning face detection guide using dlib and opencv https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/.

[tow, 2022a] (Accessed 4.4.2022a). Face recognition project https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348.

[med, 2022b] (Accessed 4.4.2022b). Face recognition with machine learning guide https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

[dat, 2022] (Accessed 4.4.2022). Facial landmark detection guide https://datahacker.rs/009-how-to-detect-facial-landmarks-using-dlib-and-opencv/.

[mtc, 2022] (Accessed 4.4.2022). Github page for mtcnn face detector implementation https://github.com/ipazc/mtcnn.

[tow, 2022b] (Accessed 4.4.2022b). Guide to create a face recognition system https://towardsdatascience.com/a-beginners-guide-to-building-your-own-face-recognition-system-to-creep-out-your-friends-df3f4c471d55.

[tow, 2022c] (Accessed 4.4.2022c). Lbph method explanation https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b.

[dnn, 2022] (Accessed 4.4.2022). Opencv dnn module guide https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/.

[fac, 2022c] (Accessed 4.4.2022c). Yolo guide in opencv https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/.

[med, 2022e] (Accessed 8.4.2022e). Object detection comparison and explanations of their evolution https://medium.com/analytics-vidhya/evolution-of-object-detection-582259d2aa9b.

[med, 2022f] (Accessed 8.4.2022f). Object detection methods comparison https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359.

[mot, 2022] (Accessed 9.4.2022). History of how ai improved computer vision https://www.motionmetrics.com/how-artificial-intelligence-revolutionized-computer-vision-a-brief-history/.

[vio, 2022] (Accessed 9.4.2022). Wiki page of viola-jones algorithm https://en.wikipedia.org/wiki/Viola/E2/80/93Jones_object_detection_framework.

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Ahonen et al., 2006] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041.

[Balaban, 2015] Balaban, S. (2015). Deep learning and face recognition: the state of the art. *Biometric and surveillance technology for human and activity identification XII*, 9457:68–75.

[Bradley and Roth, 2007] Bradley, D. and Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, 12(2):13–21.

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[Carcagnì et al., 2015] Carcagnì, P., Del Coco, M., Leo, M., and Distante, C. (2015). Facial expression recognition and histograms of oriented gradients: a comprehensive study. *SpringerPlus*, 4(1):1–25.

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Erhan et al., 2014] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154.

[Freeman and Roth, 1995] Freeman, W. T. and Roth, M. (1995). Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, volume 12, pages 296–301. Zurich, Switzerland.

[Ghorbel et al., 2020] Ghorbel, A., Aydi, W., Tajouri, I., and Masmoudi, N. (2020). Hybrid approach for face recognition from a single sample per person by combining vlc and gom. *Journal of Intelligent Systems*, 29(1):1523–1534.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Gurkan et al., 2019] Gurkan, F., Sagman, B., and Gunsel, B. (2019). Yolov3 as a deep face detector. In *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 605–609.

[Hazarika and Kumari, 2019] Hazarika, P. and Kumari, M. (2019). Evolution of modern deep learning methods of object recognition. *Res. Comput. Sci.*, 148(3):71–79.

[He et al., 2016a] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

[He et al., 2005] He, X., Yan, S., Hu, Y., Niyogi, P., and Zhang, H.-J. (2005). Face recognition using laplacianfaces. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):328–340.

[Huang et al., 2007] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[Kazemi and Sullivan, 2014] Kazemi, V. and Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874.

[Kim et al., 2020] Kim, J.-a., Sung, J.-Y., and Park, S.-h. (2020). Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4. IEEE.

[King, 2009] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758.

[King, 2015] King, D. E. (2015). Max-margin object detection. *arXiv preprint arXiv:1502.00046*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[Li et al., 2017] Li, S., Deng, W., and Du, J. (2017). Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 2584–2593. IEEE.

[Li, 2021] Li, W. (2021). Analysis of object detection performance based on faster r-cnn. In *Journal of Physics: Conference Series*, volume 1827, page 012085. IOP Publishing.

[Lienhart and Maydt, 2002] Lienhart, R. and Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing*, volume 1, pages I–I. IEEE.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

[Mizgajski and Morzy, 2019] Mizgajski, J. and Morzy, M. (2019). Affective recommender systems in online news industry: how emotions influence reading choices. *User Modeling and User-Adapted Interaction*, 29.

[Papageorgiou et al., 1998] Papageorgiou, C. P., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 555–562. IEEE.

[Phon-Amnuaisuk et al., 2018] Phon-Amnuaisuk, S., Murata, K. T., Pavarangkoon, P., Yamamoto, K., and Mizuhara, T. (2018). Exploring the applications of faster r-cnn and single-shot multi-box detection in a smart nursery domain. *arXiv preprint arXiv:1808.08675*.

[Rahmad et al., 2020] Rahmad, C., Asmara, R., Putra, D., Dharma, I., Darmono, H., and Muhiqqin, I. (2020). Comparison of viola-jones haar cascade classifier and histogram of oriented gradients (hog) for face detection. In *IOP conference series: materials science and engineering*, volume 732, page 012038. IOP Publishing.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[Rowley et al., 1998] Rowley, H. A., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38.

[Shen, 2019] Shen, X. (2019). A survey of object classification and detection based on 2d/3d data. *arXiv preprint arXiv:1905.12683*.

[Silva et al., 2015] Silva, C., Bouwmans, T., and Frélicot, C. (2015). An extended center-symmetric local binary pattern for background modeling and subtraction in videos. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP 2015*.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Suma and Raga, 2018] Suma, S. and Raga, S. (2018). Real time face recognition of human faces by using lbph and viola jones algorithm. *International Journal of Scientific Research in Computer Science and Engineering*, 6(5):6–10.

[Taigman et al., 2014] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708.

[Tan et al., 2021] Tan, L., Huangfu, T., Wu, L., and Chen, W. (2021). Comparison of retinanet, ssd, and yolo v3 for real-time pill identification. *BMC medical informatics and decision making*, 21(1):1–11.

[Tan and Le, 2019] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

[Tan and Triggs, 2010] Tan, X. and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing*, 19(6):1635–1650.

[Tayib and Jamaludin, 2016] Tayib, S. and Jamaludin, Z. (2016). An algorithm to define emotions based on facial gestures as automated input in survey instrument. 22.

[Trigueros et al., 2018] Trigueros, D. S., Meng, L., and Hartnett, M. (2018). Face recognition: From traditional to deep learning methods. *arXiv preprint arXiv:1811.00116*.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

[Wang et al., 2009] Wang, X., Han, T. X., and Yan, S. (2009). An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th international conference on computer vision*, pages 32–39. IEEE.

[Wen et al., 2016] Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer.

[Wójcik et al., 2016] Wójcik, W., Gromaszek, K., and Junisbekov, M. (2016). Face recognition: Issues, methods and alternative applications. *Face Recognition-Semisupervised Classification, Subspace Projection and Evaluation Methods*, pages 7–28.

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

[Zhang et al., 2016] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

[Zhang et al., 2017] Zhang, X., Gonnot, T., and Saniie, J. (2017). Real-time face detection and recognition in complex background. *Journal of Signal and Information Processing*, 8(2):99–112.

[Zhao et al., 2019] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232.

# Appendices

## Sample images

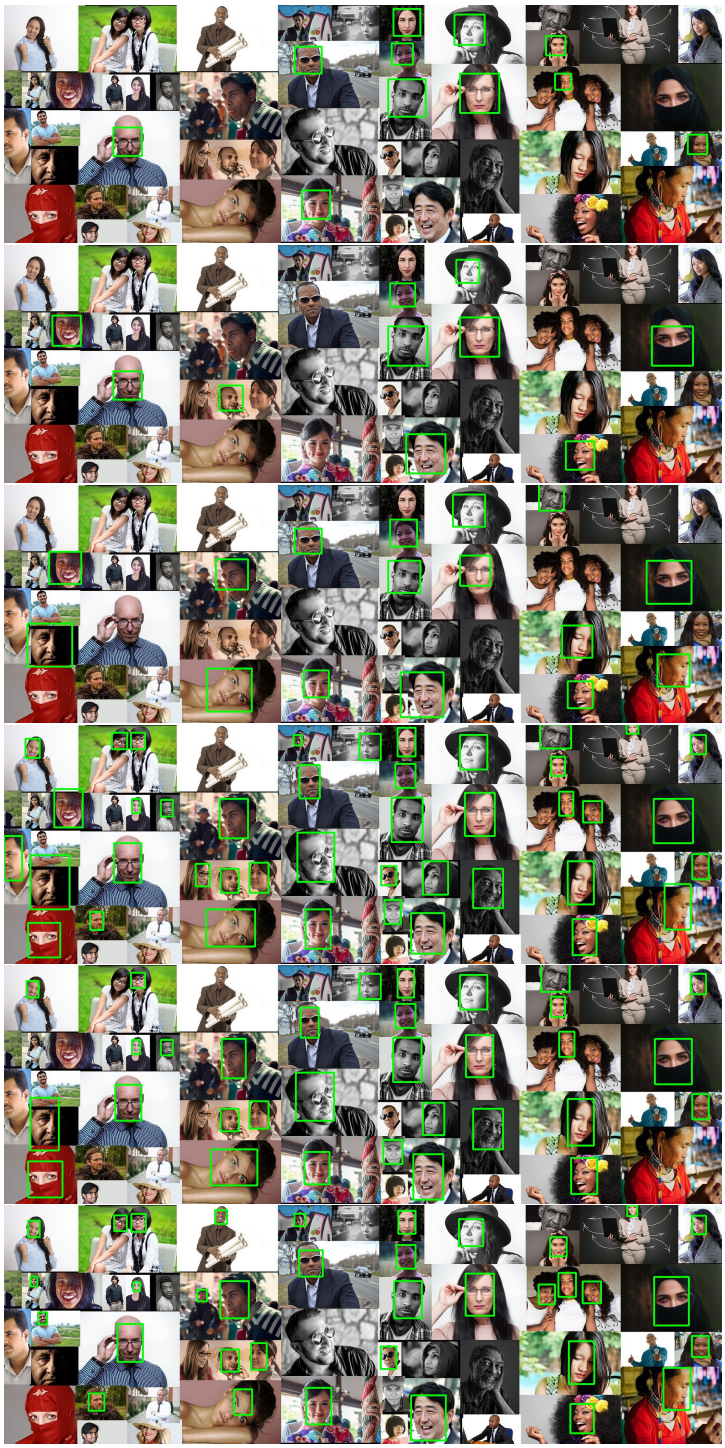


Figure 1: Top to bottom: Haar, HOG, MMOD, Caffe, Tensor, YOLO. Images from [sam, 2022a]

Figure 2: Top to bottom, left to right: Haar, HOG, MMOD, Caffe, Tensor, YOLO. Images from [sam, 2022a]

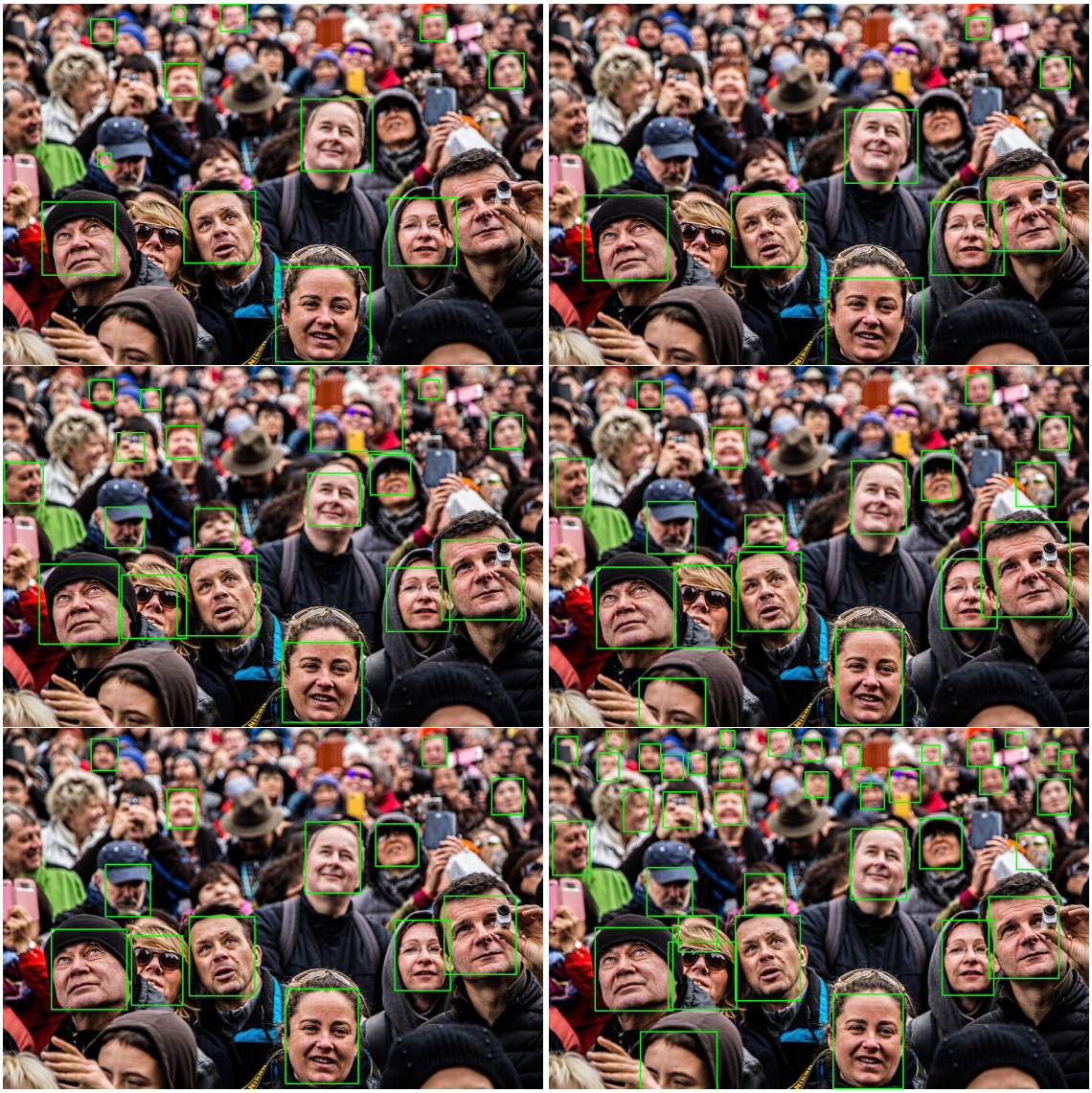


Figure 3: Top to bottom, left to right: Haar, HOG, MMOD, Caffe, Tensor, YOLO. Images from [sam, 2022a]

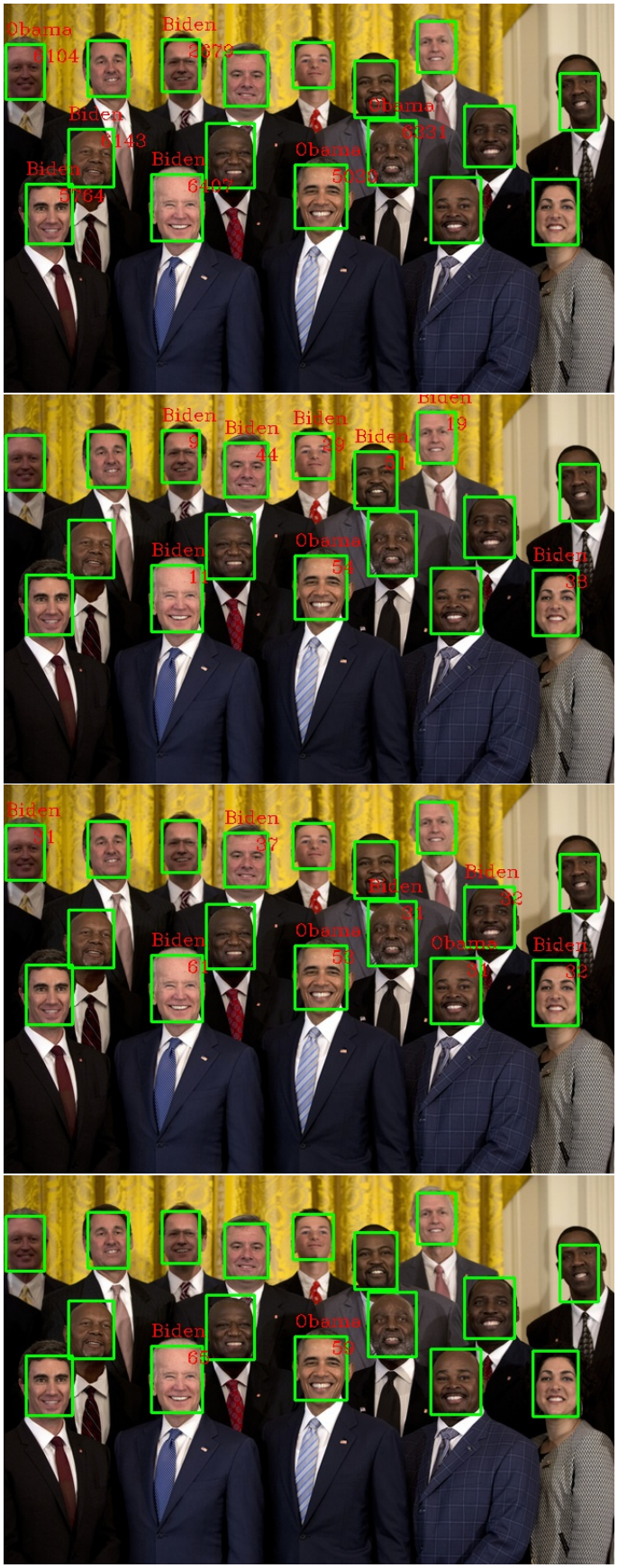
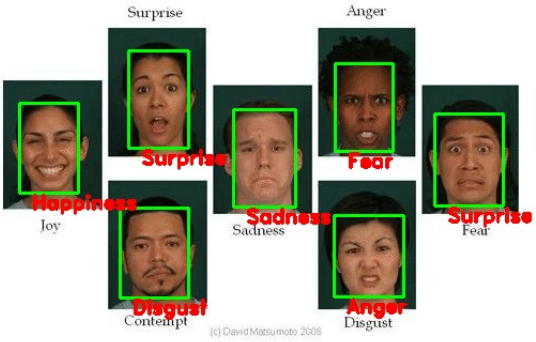
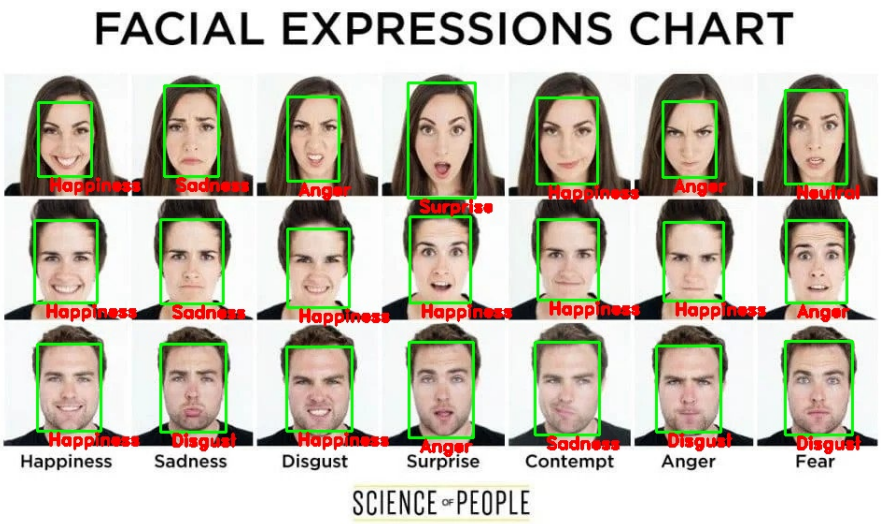



Figure 4: Top to bottom Face Recognizer methods: Eigenfaces, Fisherfaces, LBPH and Deep Learning. Image from [sam, 2022b]

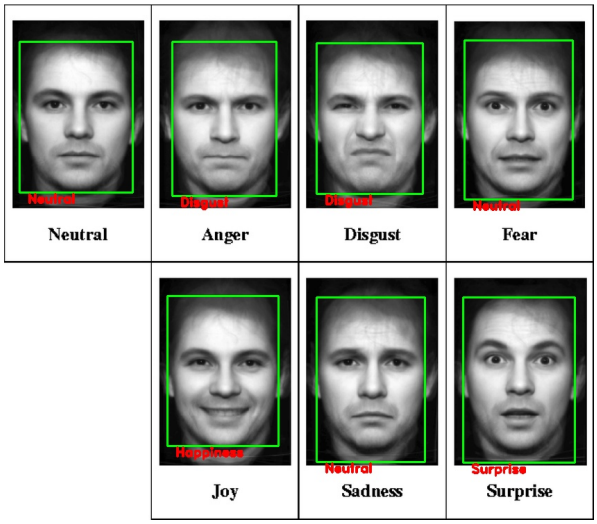




Figure 5: Emotion Recognition models used on sample images. Left to right, top to bottom, images from [sam, 2022c], [Tayib and Jamaludin, 2016], [sam, 2022d], [Mizgajski and Morzy, 2019]