Haaga-Helia

Surveying and analyzing open-source AI tools for human expression recognition.

Gianmarco Ipinze Tutuianu

Haaga-Helia University of Applied Sciences

Bachelor's Thesis

2023

Bachelor of Business Administration and ICT

Abstract

Author(s)
Ipinze Tutuianu, Gianmarco

Degree
Bachelor of Business Administration

Report/thesis title
Surveying and analyzing open-source AI tools for human face expressions recognition.

Number of pages and appendix pages
68

The ability to perceive and understand human emotions plays a crucial role in various aspects of human-computer interaction and artificial intelligence systems. In recent years, the advent of deep learning techniques, particularly Convolutional Neural Networks (CNNs), has revolutionized the field of facial emotion recognition.

This thesis explores the realm of facial emotion recognition through the lens of open-source tools and CNNs, aiming to test, analyze, train, and design an accurate and robust model capable of decoding the intricate and subtle emotional cues expressed on human face and evaluate these tools in respect of accuracy, easiness, practicality, and technical robustness.

These open-source tools and algorithms are widely used by the industry and the community to develop image classification models and real-time implementations. We are going to focus primarily on 2 open-source ready to use tools like DeepFace (Serengil & Ozpinar, 2020)., and FaceApi (Mühler, 2023), open-source frameworks such as TensorFlow and Keras where we are going to use models like Vgg16, Vgg19, MobileNet, Xception, ResNet50 among others.

We will create models from zero to test pre-trained weights efficiency, using 3 different datasets (RAF-DB, FER2013 and AffectNet) and two different class balance strategies.

The results showed that ready-to-use tools are the best for first time users, their tutorials, repositories, and guides are easy to follow and user friendly, additionally they do not require training or powerful hardware.

CNN showed the best results when using 224 x 224 resolution images, but also pretty good results when training with 48 x 48 resolutions, sacrificing some accuracy for a faster training.

Through the findings of this study, further research must be conducted to evaluate the perfect image resolution for facial expression recognition, multimodal approaches might give even better results as well.

Keywords
AI, CNN, Computer Vision, Face Detection, Face Recognition, Image Classification, Machine Learning

Table of contents

Abbreviations

FER	Facial Expressions Recognition
CNN	Convolutional Neural Network
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
TF	TensorFlow
SoTA	State of The Art
YOLO	You Only Live Once (Computer Vision algorithm)
ML	Machine Learning
AI	Artificial Intelligence
HOG	Histogram of Oriented Gradients
LBPH	Local Binary Patterns Histograms
ReLU	Rectified Linear Units Layer

1 Introduction

1.1 Emotions

Emotions, how we understand and demonstrate them, are the cornerstone of our civilization, knowing others' emotions and intentions, just by observing them and using our experience, has been fundamental for human beings' rise and development.

Being able to understand, with its limitations, feelings, emotions and sensations of others without the need for direct communication is something that we can do unconsciously using our own experience, as Dr. Ekman mentioned, different cultures have different ways of you show emotions or even name them (if there is such a word for that particular emotion), Dr. Ekman mentions that knowing and naming a specific emotion can influence how we perceive it, an example of this theory could be the expression schadenfreude (from German) which refers to the sensation of pleasure obtained by the misfortune of the enemy (Ekman, 1972).

About human expressions (facial expressions) Dr. Ekman mentions that facial expressions are both universal and cross-cultural, in other words, most of us smile when faced with a situation that makes us happy or joyful, that is a characteristic that we could call as "universal" but the intensity of this expression or the way in which we show it can vary between different cultures (Ekman, 1972).

An example of this was the experiment carried out with Japanese and North Americans students when showing them videos that could be considered unpleasant, the two groups showed similar intensity of emotions when they watched the videos in private (they were not aware of a camera recording), while when they were aware of being observed, the Japanese, but not the Americans, showed different gestures and facial expressions, the Japanese masked feelings of disgust and fear with a smile, while the Americans acted in a similar way in both situations (Ekman, 1972).

These differences make detecting emotions in faces an extremely complex task, since the intensity of the emotion and the quality of the dataset to use will mean how accurate our model can be when detecting what emotion is being shown.

1.2 Research Background

Artificial intelligence (AI) is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings (Copeland, 2023). The term is also applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as reasoning, learning, understanding and problem-solving (Copeland, 2023).

AI can be classified into two types: narrow AI and general AI. Narrow AI is focused on performing specific tasks such as playing chess or recognizing faces. General AI is a theoretical form of AI where a machine would have an intelligence equal to humans; it would have a self-aware consciousness that can solve problems, learn, and plan.

The history of AI can be traced back to ancient times when myths and stories depicted artificial beings endowed with intelligence or consciousness. The modern field of AI was founded in 1956 at a conference at Dartmouth College where John McCarthy coined the term "artificial intelligence" (*What Is Artificial Intelligence (AI) ?* | *IBM*, 2023). Since then, AI has made significant progress in various domains such as natural language processing, computer vision, robotics, machine learning and expert systems. Some of the milestones in AI include Alan Turing's test for machine intelligence (Turing, 2012), John Hopfield's neural network model for associative memory (Krotov & Hopfield, 2016), David Rumelhart's backpropagation algorithm for training neural networks (Rumelhart et al., 1986), IBM's Deep Blue chess-playing computer that defeated Garry Kasparov (Campbell et al., 2002), Google's AlphaGo program that beat Lee Sedol at Go (Silver et al., 2016), etc.

We can measure AI importance based on its many applications and benefits for humanity and society. For example, AI can help improve healthcare by diagnosing diseases, recommending treatments, and discovering new drugs. AI can also enhance education by providing personalized learning experiences and feedback. AI can also improve security by detecting threats and preventing cyberattacks. Moreover, AI can also create new opportunities for innovation, creativity, and entertainment by generating art, music, and games. However, AI also poses some challenges and risks such as ethical issues, social impacts, and existential threats. Therefore, it is essential to develop responsible and trustworthy AI that aligns with human values and goals.

AI is closely related to computer vision because both fields aim to enable machines to perceive and understand information. Computer vision is a subfield of AI that focuses on processing images and videos using mathematical techniques such as convolutional neural networks (CNNs) (LeCun et al., 1998). Computer vision can help improve various tasks

such as face recognition (Viola & Jones, 2004), object detection (Ren et al., 2017), scene understanding (Zhou et al., 2017), image generation (Goodfellow et al., 2020), etc. Computer vision can also benefit from other subfields of AI such as natural language processing (NLP) by enabling multimodal communication between machines and humans using text and images (Vinyals et al., 2015).

It is important to learn from them and know how can be used for our advantage, how we can create our own algorithms to perform tasks in a more efficient way that many humans can.

When we talk about artificial intelligence and its applications, we can see the range of possibilities is complex and full of different variations. Among all of these, the field that this project seeks to explore is Computer Vision.

1.3 Computer Vision

Computer vision is an interdisciplinary field that deals with how computers can be trained to gain high-level understanding from digital images or videos. From an engineering perspective, it seeks to automate tasks that the human visual system can do.

Scientists and engineers have been trying to develop ways for machines to see and understand visual data for about 60 years. Experimentation began in 1959 when neurophysiologists showed a cat an array of images, attempting to correlate a response in its brain (Lotlikar, 2022).

The systems are made up of a photographic or video camera and specialized software that identifies and classifies objects. They can analyze images (photos, images, videos, barcodes), as well as faces, image features and face expressions.

To teach a computer to “see”, machine learning technologies are used, and a lot of data is collected that makes it possible to highlight features and combinations of features to further identify similar objects. (RecFaces, 2022)

Within computer vision we can find different fields of application such as pattern recognition, image analysis, image classification, object detection, object tracking, content-based image retrieval among many others. (IBM, 2022).

1.4 Challenges

Some challenges faced by computer vision include inadequate hardware, poor data quality, weak planning for model development and time (Garanhel, 2023).

One reason why computer vision is so challenging is that when machines look at an image to encode it, they are not able to see them and understand them as humans do, but they do it as numbers representing the pixels. On the other hand, the human brain perceives photos as objects in a highly visual and intuitive manner. For machines, it is difficult to process all that data when training a computer vision model.

Collecting relevant and sufficient data also has various challenges. These challenges can lead to a lack of training data. For example, gathering medical data is a challenge for data annotators, this is because of the patient's privacy, the lack of good quality data, experience and more (Klinger, 2022).

Human bias is always a problem, being positive or negative, we should be aware of this when we collect data, is normal to see datasets with some lack of representation related to human varieties, is important then to focus on this problem and address it when we work with images, is necessary to get as much as possible data variation to create an accurate model.

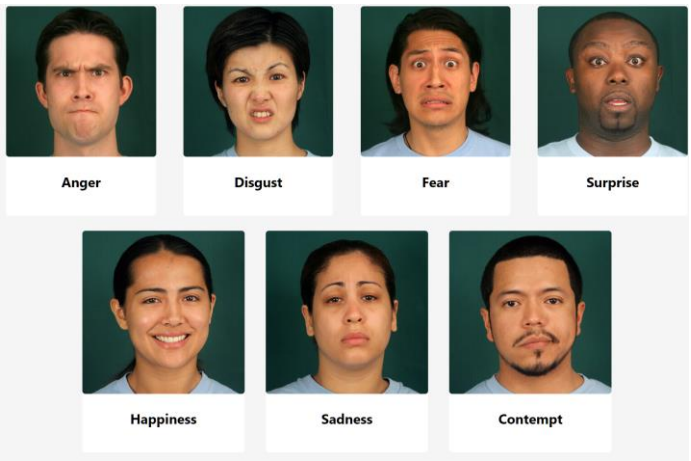

Fig 1: 7 Universal facial expressions proposed by Dr. Paul Ekman (Paul Ekman Group, 2022).

When we analyze Facial emotion recognition (FER), we can divide it into 3 steps: face detection, face expression detection and emotion classification (fig 1).

Fig 2: Steps of Facial Emotion Recognition (Vemou & Horvath, 2021)

Emotion detection is based on the analysis of facial landmarks positions (eyes, eyebrows, mouth) Furthermore, in videos, changes in those positions are also analyzed, to identify contractions in a group of facial muscles (Ko, 2018).

To teach a computer to detect, recognize and classify faces and expressions we should use tools such as face detection, feature extractors, data augmentation, and more.

This project is going to focus on open-source tools, but additionally, we will be using datasets to train out different models to perform face emotion recognition, companies such as Facebook and Google had developed open-source tools as well, using their vast amount of data, to recognize face emotions.

This project will be testing open-source tools such as DeepFace (Serengil & Ozpinar, 2020) ,OpenCV (Intel), Mediapipe (Google), and different CNN architectures using TensorFlow as framework, in order to analyze which one is the most accurate based on the same training flow, the idea of this project is to analyze the available tools, use train the needed model to perform FER and compare the results, aiming to obtain the best model.

2 Research Phenomenon and Goal

Facial Emotions Recognition (FER) is a growing field within Computer Vision, is possible to apply FER solutions to different fields such as Marketing and business with customer emotion analysis, in healthcare, helping doctors to detect autism in early stages, predict psychopathic disorders or depression, in Employment by helping recruiters to identify uninterested candidates and many more possible applications in different fields, is even possible to develop solutions, such as music recommendation tool, marketing campaign design and more.

The goal of this project is surveying and analyzing open-source AI tools for human face expressions recognition.

Is important for students to understand the possibilities and for universities and institutes being able to analyze the available open-source technologies and tools to create action plans to introduce the topic to new students and expand their possibilities and knowledge.

2.1 Why is worth doing the research.

Computer Vision and FER are growing technologies, *State of the Art,* which means that there is still a vast field to explore, learning a technology while is relevant and is still under development will help students to be ready for employment as soon as they end the career, this will be beneficial for everyone while it promotes research and education quality at the same time.

The information collected in this research could help educational centers to fine-tune their educational plans to provide the latest knowledge related to computer vision, Machine Learning and Artificial Intelligence.

2.2 Research Objectives

The goal of this project is to study, test and analyze different algorithms for facial emotion recognition (FER). Each algorithm is going to be trained with the same dataset, and after the training, a different dataset is going to be used to test. Finally, the results are going to be compared using the following parameters:

- Accuracy
- Easiness
- Practicality
- Technical robustness

The idea is to compare the results based on these 4 concepts and point them to find the best one in average.

As it was mentioned before, the goal of this project is to analyze only open-source tools, this is due to its availability, the community, and the cost limitations.

2.3 Research Questions

How good are the open-source AI tools for human face expressions recognition.

2.4 Scope of the Project

The analysis of facial expressions by computer requires a set of steps to follow, these steps are necessary to be able to obtain good results, since they will provide us with both the dataset to feed the algorithm and the possibility of comparing the results.

It is therefore necessary to present the steps to follow to have a general understanding of the process and that the steps performed can be repeated for the purpose of future tests.

To start with the recognition of facial expressions, it is necessary to first determine the dataset, possibly this is the step that takes the most time due to how complex it can be to obtain a quality dataset as needed.

Once the dataset is obtained, it is advisable to pre-process it to clean the images that are considered unnecessary, adjust sizes, and make corrections, apply the same data augmentation, and train the models, when they are necessary, using the same steps.

The complexity of the project lies in being able to find open-source tools that allow us to perform the same function (recognize facial expressions) using the same training dataset, for this there are different approaches such as PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis), Neural Networks and others. This project aims to be able to explain these concepts in a concise way, and to compare the tools accuracy and robustness we will apply confusion matrix and class prediction vs class truth.

2.5 Definitions

Computer Vision – Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image. (What Is Computer Vision? | IBM, 2023.)

Facial Expression Recognition - Facial Emotion Recognition is a technology used for analyzing sentiments by different sources, such as pictures and videos. It belongs to the family of technologies often referred to as “affective computing”, a multidisciplinary field of research on computer’s capabilities to recognize and interpret human emotions and affective states and it often builds on Artificial Intelligence technologies. (Vemou & Horvath, 2021)

CNN – A convolutional neural network (CNN) is a type of deep learning neural network designed for processing structured arrays of data such as images. CNNs are widely used in computer vision and have become the state of the art for many visual applications such as image classification. They are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. CNNs have three main types of layers: convolutional layer, pooling layer, and fully connected (FC) layer.

OpenCV - OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code. (OpenCV, 2022)

TensorFlow - TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning models and algorithms (aka neural networks) and makes them useful by way of common programmatic metaphors. It uses Python or JavaScript to provide a convenient front-end API for building applications, while executing those applications in high-performance C++.

Keras - Keras is an open-source high-level neural network library, which is written in Python and is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. Keras is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks.

Keras is an API designed for human beings, not machines. It follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

YOLO - You Only Look Once (YOLO) is a popular model architecture and object detection algorithms. It uses one of the best neural network architectures to produce high accuracy and overall processing speed, which is the main reason for its popularity. If we search Google for object detection algorithms, the first result will be related to the YOLO model.

YOLO algorithm aims to predict a class of an object and the bounding box that defines the object location on the input image. It recognizes each bounding box using four numbers:

- Center of the bounding box ((b_{x}, b_{y}))
- Width of the box (b_{w})
- Height of the box (b_{h})

FaceApi – It is a JavaScript library created by Vincent Mühler., to detect faces via browser. It is built over tensorflow.js core API. It supports Face Detection, Face Recognition, Face Expression, Age, and Gender Detection (Face-Api.Js, 2023)

DeepFace - DeepFace is a lightweight face recognition and facial attribute analysis (age, gender, emotion, and race) framework for python.

It is a hybrid face recognition framework wrapping state-of-the-art models: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace and Dlib. The library is mainly based on Keras and TensorFlow. (Serengil & Ozpinar, 2020).

3 Theorical Background

Face Emotion recognition is a complex task which is possible to subdivide into smaller tasks which are necessary to achieve the goal, face emotion recognition.

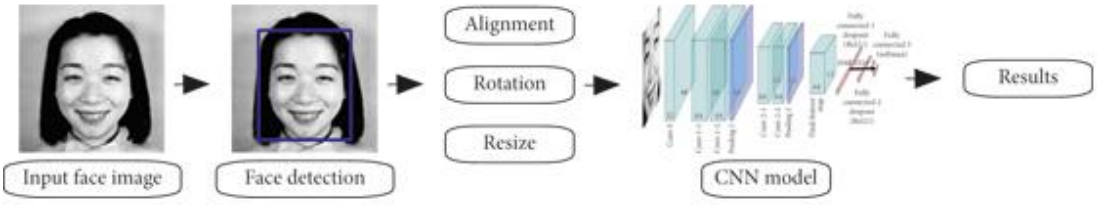

Fig 3: Face Emotion Recognition process (Wang et al., 2020)

3.1 Input Image

Input image refers to the digital image that is fed into the system for processing. This image typically contains one or more faces and is used by the system to detect and analyze facial expressions to recognize emotions.

3.2 Pre-Processing Images

When we are dealing with data, is important to understand that we need to have the best possible data we can get, this means that, once we are able to work with our desire data (in the case of this project are images) we should realize the tedious job of analyzing it and delete what is called “noise”, basically garbage, this garbage could be images that we don’t need for our training or testing, too big images with too many faces, bad quality images, and more.

In this sense, when we pre-process, we should understand that there are 2 different ways to do, which normally we do both, these are “on the fly” or, as the concept implies, before the main process in-situ, which means reviewing each image and delete the garbage and apply the desired techniques over them (face extraction, for example), some of these pre-processing steps are, and not limited to, resizing, orienting, and color corrections (Nelson, 2020)

Preprocessing images is important due to different factors, for example, at reducing the model training time, by reducing the image sizes, also, CNN models require that all images are the same size arrays, if images are not the same size, this might cause problems and the model may not perform as expected (a common problem using TensorFlow library, for example), as an example, working with images with irregular sizes and significant bigger sizes that the desired (if we try to resize image to 48 x 48 while our dataset

contains images of 650 x 650, for example) can multiply the training time in more than 4 times, applying a image resize in-situ could be a good solution for this problem.

Here I will explain one of the most common pre-processing steps we can apply to our models.

3.2.1 Resizing

Resizing, as the name suggest, is changing the size of the images, to do this we should consider different aspects, such as:

- CNN Model architecture

  Not all the models accept all the input sizes, for example ResNet model, as the name suggest, reduce the input sizes, this means that if the input (image resolution) is too small, the model won't be able to perform.

- How important is image resolution for us.

  When we aim to design and train a CNN model, we need to work with different external factors and create a strategy for them, in this case, we need to decide if we aim to have a good accuracy but slow training process (bigger image resolution) or fast training and less good training accuracy (smaller resolution images) this is something we should be aware beforehand.

  A study from 2020 showed that the maximum performance of some of the most known CNN architectures is reached with images between 256 x 256 and 448 x 448 (Sabottke & Spieler, 2020).

- Computational resources

  We need to know, before training a CNN model, that these require an important computational power, it is possible to train models using our CPU (in a laptop for example) but the training times are going to be seriously big, this due to the processing cores, which are around 4 to 8 in a CPU, but more than 3000 in an entrance GPU nowadays, that is why GPUs are used for this task, the amount of processing cores in a gpus accelerates significantly the training processes.

Something to consider when we resize images it's to know that many architectures use square like images, and force images to resizing might end stretching it dimensions to fit to be square or keeping the aspect ratio constant and filling the newly created space with new pixels.
The recommendation is to not always resize by scaling, reflected image content also works well and down sampling is the safest (Nelson, 2020).

3.2.2 Channels (RGB or Grayscale)

Images, normally are represented in color, RGB (Red, Green, Blue) this means that a pixel (which values are represented in an scale from 0-255) is a mix of 3 layers of RGB pixel, this is a lot of data to process for the majority of computers, and if it's not a problem, then is still too much extra job to handle, to solve this "problem" the best solution we can implement is to convert our RGB image to grayscale.

Grayscale calculates the pixel value (from the 3 layers) and returns a mean value equivalent to the scale of gray, where 0 is the absence of light (intensity) which means black and 255 is white.

Fig 4: Example of RBG to Grayscale (Pytorch Forum, 2019).

If we consider that the color is not significant for our training process, then could be a good idea to transform our images to grayscale, we might lose some information related to contrast, but we can accelerate the training process, this is a tradeoff we must test and manage (Di Palo, 2018)

3.2.3 Data Normalization

Data normalization is a process that changes the pixels intensity range values in an image. It's also known as contrast stretching or histogram stretching.

Normalization helps to improve the quality and consistency of data by re-scaling it to have a standard deviation of 1 (unit variance) and a mean of 0. This is useful when you have data from different formats (or datasets), and you want to normalize all of them so you can apply the same algorithms over them.

An example of this could be having an image with pixel intensity values ranging from 0 to 255, normalization would involve subtracting the mean pixel intensity value from each pixel and then dividing by the standard deviation of the pixel intensity values. This would result in a new image with pixel intensity values that have a mean of 0 and a standard deviation of 1 (Riva, 2023).

This can help improve the quality and consistency of data by making it easier for algorithms to process and analyze the image.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Fig. 5: Scaling to range formula

It’s important to know that, when we work with pre-trained models, we need to know how the model was trained and designed, some models use a 1/255 division (0 and 1) while other models use a 1/127.5 division (-1 and 1). We should be aware of this before training our model because it might affect performance.

3.2.4 Noise Removal (De-noising)

Image denoising or image noise removal, is the process of removing noise from an image. Noise can be introduced into an image during the image acquisition process or during transmission. The goal of denoising is to restore the true image while preserving important details.

There are several techniques for denoising images such as spatial filtering, temporal accumulation, and machine learning and deep learning reconstruction.

The choice of technique depends on the type and amount of noise present in the image, it has been proved its efficiency increasing the model's accuracy and performance (Ilesanmi & Ilesanmi, 2021).

Fig 6: Denoising an image example (Saxena & Kourav, 2014)

3.2.5 Face Cropping

Face cropping is a step-in image pre-processing where faces are detected and then cropped from an image. This can be done using a pre-trained Haar Cascade model to detect faces from the image. The goal of face cropping is to isolate the face from the rest of the image for further processing.

Face cropping can be useful for tasks such as face emotion recognition where it's important to focus on just the face. By removing other parts of the image, it can improve accuracy and reduce noise. Face cropping is often used in combination with other pre-processing steps such as face alignment.

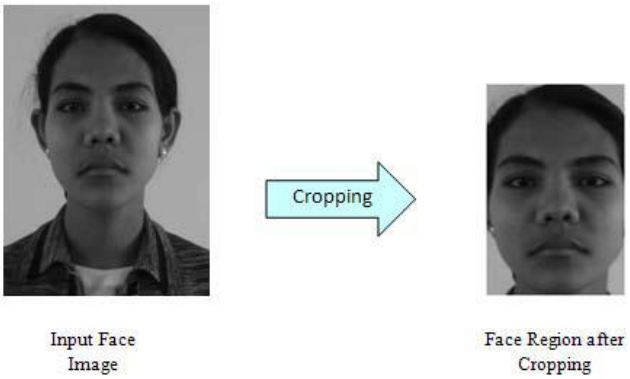

Fig 7: Face Cropping example (Saxena & Kourav, 2014)

3.2.6 Face Alignment

Face alignment is a step-in image pre-processing where faces are transformed into a common coordinate system. This can be done by using landmark prediction models that find points on the face such as the nose, eyes, and mouth. The goal of face alignment is to warp faces so that they have the same alignment for better accuracy.

Face alignment is important for tasks such as face recognition where it's important for all faces to be aligned in a common coordinate system. By aligning faces, it can improve accuracy and reduce noise. Face alignment is often used in combination with other pre-processing steps such as face cropping (Gu & Kanade, 2008).

Fig 8: Face alignment example (Serengil & Ozpinar, 2020)

3.2.7 Data Augmentation

Data Augmentation is manipulation forms for our dataset, in other words it creates (or modifies) new data based on that already existing one.

Data augmentation is useful to simulate possible scenarios that are not contemplated in our dataset, an example of this could be a face dataset, normally people might not be hanging upside down, so we don't consider that scenario, but maybe our model is going to be used to frame faces from people on roller coasters, and people on roller-coasters could be upside down, so this means that our model might not work as good as it could be, to solve this problem we use data-augmentation algorithms to create image of our faces dataset turn in 180° degrees, so we can represent faces upside down (Medium, 2020)

Fig 9: Data Augmentation example (Manmeet Singh, 2020)

3.3 Facial Expression Recognition

To recognize emotion in faces, we must first teach the computer to look for them in inputs, which could bel images, videos or by using cameras.

To train our algorithm it is necessary to have, as mentioned above, a dataset large enough to provide the algorithm with different features related to faces, if the dataset is not large and complex enough, then our algorithm will fall into a problem called "overfitting", this means that the algorithm has been trained so well with the provided dataset that it will be difficult for it to recognize faces other than those included in the training (CFI, 2022).

Once we have trained our algorithm, the FER process can be simplified as follows:

1. We must be able to identify face(s) in the image,
2. The algorithm we trained must be able to extract features from our input.
3. The algorithm will compare the extracted features with the trained ones to find similar patterns.
4. In case of finding similar patterns, the algorithm will then identify the face expression and return a positive value based on the argmax value, which means the closest value to our input.

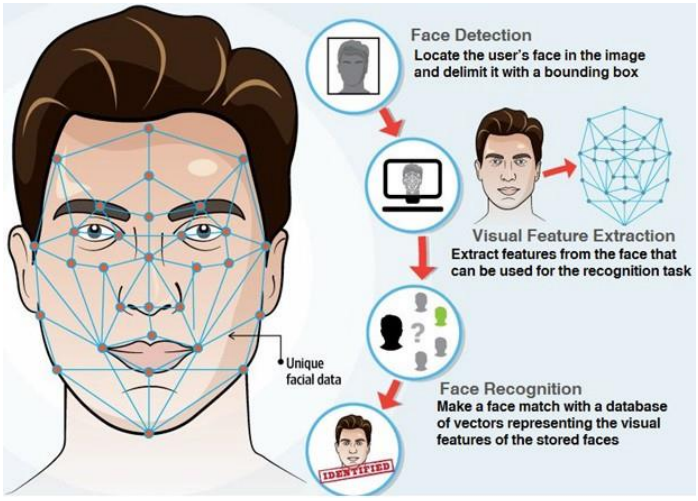

Fig 10: Face Recognition process (Fenjiro, 2019)

3.4 Facial Expression Recognition process

As was mentioned before, the process for facial emotion recognition is complex and must follow certain steps before it can be accomplished.

In this chapter we are going to explain this process, the steps involved and technicalities that might help us to achieve our goal.

3.4.1 Face Detection

Face detection is an artificial intelligence-based computer technology used to find and identify human faces in digital images. It can be applied to various fields such as security, biometrics, law enforcement, entertainment, and personal safety.

Face detection can be regarded as a specific case of object-class detection where the task is to find the locations and sizes of all objects in an image that belongs to a given class. There are several algorithms for face detection such as Haar Cascades and Deep Learning-based methods. The choice of algorithm depends on factors such as accuracy, speed, and robustness (Yan et al., 2014).

3.4.2 Face Detection Methods

There are different approaches to detect faces in computer vision, in this project we will focus on those with more information available, a more active community and more bibliography.
Among these methods we can find:

3.4.2.1 Viola Jones

The Viola-Jones algorithm, named after two computer researchers who proposed the method in 2001, Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”, is a, still despite the time, a powerful method to detect face in real time.

The algorithm, which uses grayscale images, divides the image into small “windows” or subregions and tries to find a face by comparing the trained data with the current input. These subregions use Haar like features to detect faces (Great Learning Team, 2022).

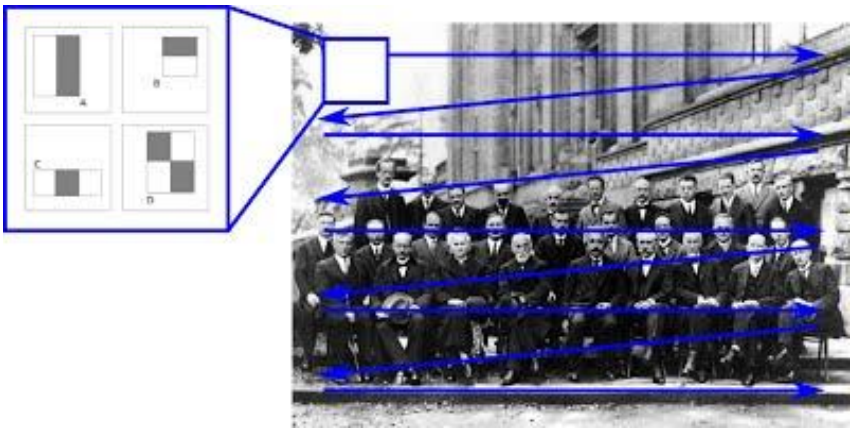

Fig 11: Viola Jones detection process (Great Learning Team, 2022)

3.4.2.2 Haar-Cascade

Haar-Cascade is an object detection algorithm that uses edge, lines, and four-side haar-features to analyze the pixel differences in images (Viola & Jones, 2004) in this case, face detections.

To simplify, we can understand that all human faces share universal similarities, such as eyebrows, eyes, mouth, regions that are, normally, darker than their neighbor pixels (an eyebrow is darker than the skin around it, for example) and to analyze these differences, Haar-like features are efficient.

Haar like features analyze by mathematic methods the pixel color intensity comparing it to its neighbor and assigning values based on its own value plus a summatory of the pixels around it, this is what is called “Integral Image” and allows the algorithm to work faster.

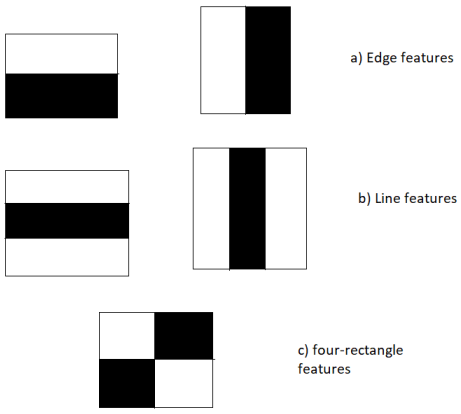

Fig 12: Haar Features (Sachin, 2023)

Currently is possible to find very efficient pre-trained Haar models on web repositories such as GitHub but is possible to train our own Haar-Cascade algorithm using GUI tools such as Cascade-Trainer GUI or also by using OpenCV framework.

It's important to know that if we want to train a Haar-Cascade detector we need to have several positive images and negatives as well. Positive images are those "things" we want to detect, in our case faces, and negative images are all those things we don't want to detect, and based on the community recommendations, we should try to review our negative dataset to not have any positive element on them.

Fig 13: Example of positive and negative images for humans(Phuc et al., 2019)

3.4.2.3 HOG (Histogram of Oriented Gradients)

Histogram of Oriented Gradients (HOG) is a feature descriptor and it's used in computer vision and image processing for object detection.

HOG works by analyzing a pixel and its neighbors. The main idea, as the name HOG implies, is to analyze the edges of the elements in an image to plot the orientation and gradient of them.

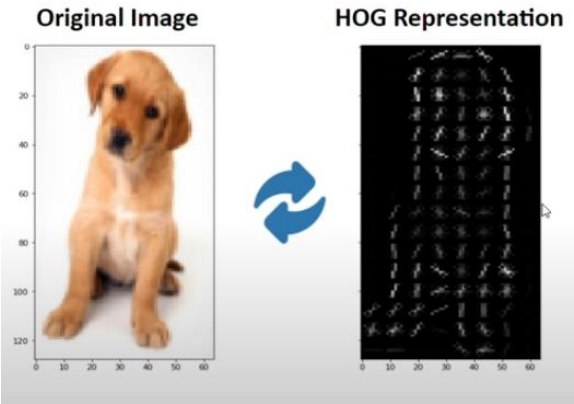

Fig 14: Image vs HOG Representation (A. Singh, 2019)

HOG decomposes an image into small, squared cells, computes a histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and returns a descriptor for each cell 2. The HOG descriptor focuses on the structure or shape of an object and uses both the magnitude and angle of the gradient to compute features (Skillcate AI, 2022).

3.4.2.4 MTCNN

Multi-task Cascade Convolutional Neural Network is one of the most powerful and latest ways for face detection. This model uses three convolutional networks (P-Net, R-Net, O-Net) (Gradilla, 2020).

This tool detects faces based on a stage process, where the first stage is:

RESIZING - In this step the algorithm creates an image size pyramid, which is going to work as the input for the next 3 Convolutional Stages.

The image pyramid is created artificially to teach the algorithm to detect different face sizes within the image.

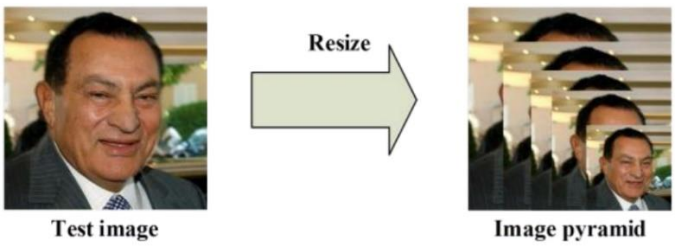

Fig 15: MTCNN Image resize (Zhang et al., 2016)

Stage 1 - P-Net – In this stage the algorithm, with a 12x12 kernel, is going to run through the images, scanning for faces, and the result is going to pass to P-Net (Proposal Network) for this to make a coordinate box if it detects a face, with this proposed candidate, the algorithm process to apply Non-Maximum Suppression (NMS) to merge overlapped candidates. (Medium, 2020)

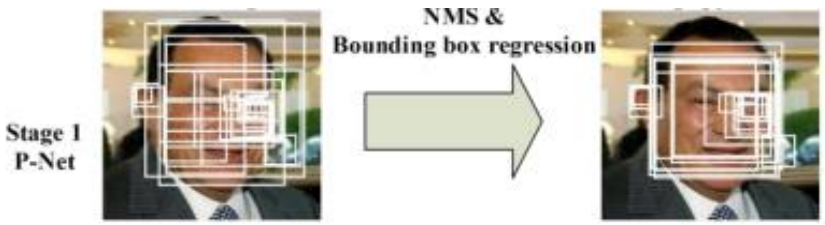

Fig 16: MTCNN P-Net (Zhang et al., 2016)

NMS – Non-Maximum Suppression is a technique used in computer vision. It's an algorithm that helps to select on entity (bounding box) out of many overlapping entities, we can choose the selection criteria to get the expected result, these criteria are normally some forms of probability or some form of overlap measure (intersection over union) (LearnOpenCV, 2021).

Fig 17: NMS application (Prakash, 2021)

Stage 2 – R-Net – This stage, called Refine Network, this stage is fed with the previous stage (P-Net) and proceeds to reject many false candidates, calibrates with bounding boxes regression, and performs NMS candidate merge.

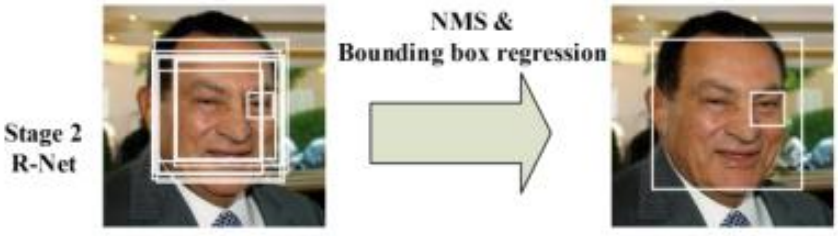

Fig 18: R-Net application (Zhang et al., 2016)

Stage 3 – O- Net – The third stage, the Output Network aims to describe the face in more detail, will output five facial landmarks position.

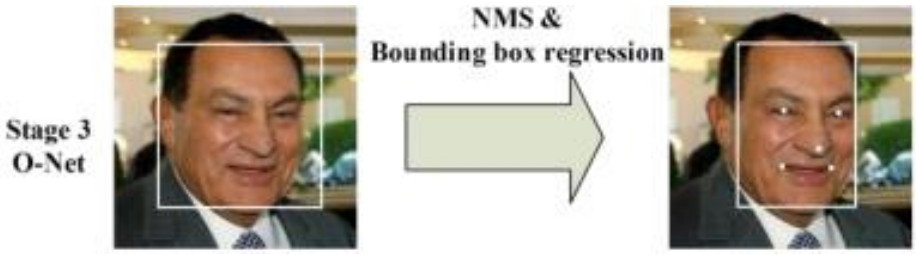

Fig 19: O-Net application (Zhang et al., 2016)

One of the benefits that this model (MTCNN) is that it uses less computational complexity, which means that with less “power” it obtains high accuracy results, this due to its convolutional nature, where in the first 2 stages it’s cleaning the path for the last stage where it finally starts to detect facial landmarks, making it faster than previous methods (Zhang et al., 2016).

3.4.2.5 YOLO (You Only Live Once)

YOLO is a SoTA computer vision model for object detection and classification, currently the YOLOv8 version is available with a deeper and easier integration making the learning process faster and more user-friendly, is an easy to learn algorithm for new users.

Unlike other methods where images are scanned with a sliding window, in YOLO the whole image is passed into a convolutional neural network and predicts the output in one pass.

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds.

YOLO can only predict a limited number of bounding boxes per grid cell, 2 in the original research paper. And though that number can be increased, only one class prediction can

be made per cell, limiting the detections when multiple objects appear in a single grid cell (Redmon & Farhadi, 2018)

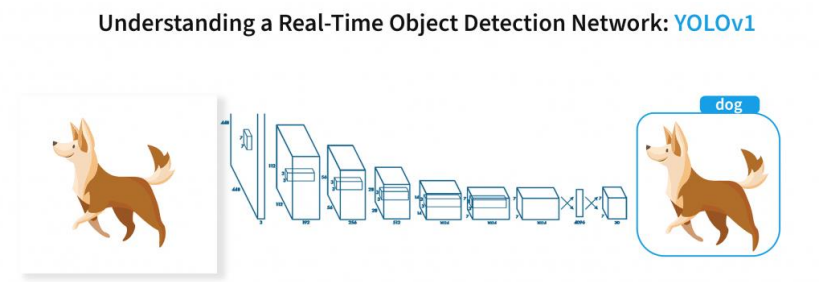

Fig 20: YOLO workflow (Sharma, 2022)

3.4.3 Face Expression Recognition Methods

There are different approaches for FER, statistically, mathematically, pattern based, CNN and more, in this chapter we are going to explain the most representative approaches, such as Eigenfaces, LBPH, Fisherfaces, DL and CNN (different models).

Eigenface: This method uses Principal Component Analysis (PCA) to extract the most important features of a face and create a feature vector, which can then be compared to other feature vectors to perform recognition.

Fisherface: Like Eigenface, Fisherface uses Linear Discriminant Analysis (LDA) to extract the most important features of a face and create a feature vector.

Local Binary Patterns (LBP): This method uses the distribution of texture in a face image to create a histogram of patterns, which can then be compared to other histograms to perform recognition.

Deep Learning-based methods: This method uses Convolutional Neural Networks (CNNs) to perform facial recognition. CNNs can learn to recognize faces directly from image data and have been shown to outperform other methods on many benchmark datasets.

Each of these methods has its own strengths and weaknesses, and the choice of method will depend on the specific use case and requirements of the recognition system.

3.4.3.1 Eigenfaces

Eigenfaces is a statistically based face detection method (Turk & Pentland, 1991). The technique is based on the extraction of the main component, this component is what can be considered as the most significant element within the image.

This algorithm is based on 2 stages, Training and Classification.

In training, a data set is provided (in this case images) and the algorithm proceeds to extract the principal components, creating a faction space (eigenspaces). The eigenspaces is the matrix formed by a series of eigenvectors that contains the information on the variation of the gray values of each pixel of the set of images used (

For the training, it is important to consider that the images must, as far as possible, meet certain characteristics that will help us obtain better results, light normalization and image standardization are important for the training set because this is going to help us to create a more accurate image matrix.

For the Classification, we select and input (this could be an image or video) and the algorithm proceeds to compare the selected input over the trained eigenspaces, using the Euclidean distance (the distance between the eigenvectors), is going to tell us which “emotion” corresponds in a closer range, to our new input. (TowardDataScience, 2018)

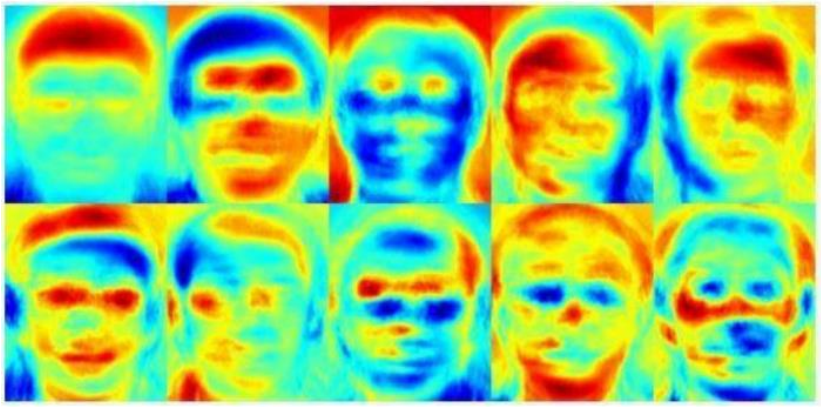

Fig 21: Eigenfaces PCA example (Nagel et al., 2002)

3.4.3.2 LBPH (Local Binary Patterns Histogram)

LBPH is an algorithm that works by calculating the pixel values using the “neighbors” are reference, giving as result a binary number.

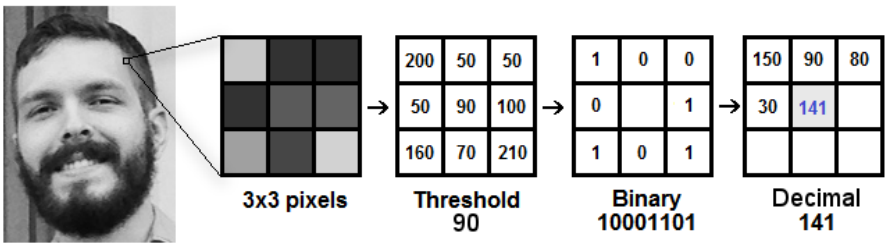

Fig 22: LBPH pixel matrix process (P. Singh, 2021)

In this process, the image is being run by a window (3x3 windows) which is going to detect the pixel grayscale value (threshold as an example) and, once the threshold values are recognized it uses a simple mathematic formula to assign the binaries (P. Singh, 2021)

The rule says:

- We can do the assigning process in any order (clockwise, anticlockwise) and we can start from any neighbor we decide.
- If the neighbor’s value is bigger or the same as the central value, we assign a 1, if the neighbor’s value is less than the central value, we assign a 0.
- Once we have assigned the binaries, we should apply the formula to assign the pixel value, this value might go from 0 to 255.
- This new pixel value not only tells us about the pixel value, but also has information related to its neighbors.

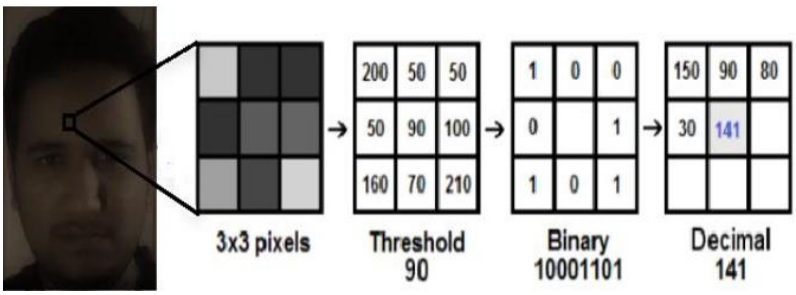

Fig. 23: LBPH pixel conversion (Jagtap et al., 2019)

3.4.3.3 Fisherfaces

Fisherfaces is a technique in computer vision used for face recognition. It is based on the Fisher Linear Discriminant Analysis (FLDA) algorithm, which seeks to project the original image data onto a lower-dimensional subspace while maximizing the between-class scatter and minimizing the within-class scatter. In the context of face recognition, each class represents a unique person. The resulting Fisherfaces are a set of orthogonal eigenvectors that best represent the unique facial features of each person and can be used for face recognition by comparing the similarity between the projected test image and the Fisherfaces of each person in the database. The Fisherfaces method has been shown to be more robust to changes in lighting, pose and expression than other traditional face recognition methods, making it a useful tool for facial recognition applications (Anggo & La Arapu, 2018)

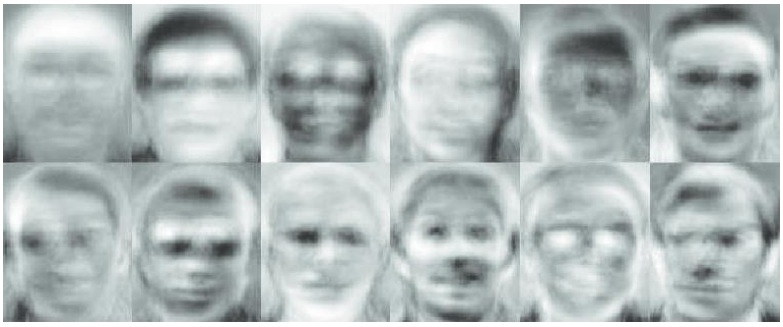

Fig 24: Fisherface representation (Chihaoui et al., 2016)

3.4.4 Deep Neural Networks (DNN)

Deep Neural Networks (DNN) are a family of neural networks that contain multiple layers and neurons, which allow them to process complex data (like images, audio, and written language), it's called "deep neural networks" because it has more than 2 layers (deep), imitates (or tries to) how human neuron works, being interconnected as a network (Johnson, 2020)

DNNs have been successful in achieving state-of-the-art performance in many tasks and have become an important tool in artificial intelligence and machine learning.

In DNN (as in Machine Learning and AI) we have 2 big groups when we talk about data classification, these are supervised and unsupervised, the first one relies on the labels, in this case the classes or the emotion, while unsupervised works with unlabeled data

Among the DNN, we can differentiate them by usage and specialization, these are:

- MLP (Multi-Layer Perceptron)
- Modular Neural Networks
- CNN (Convolutional Neural Networks)
- RNN (Recurrent Neural Networks)
- RBF (Radial Basis Function Neural Networks).

These are the most known DNN, but for the goal of this project, we are going to focus on CNN and Modular Neural Networks.

3.4.4.1 Modular Neural Networks

A modular neural network is a type of artificial neural network that consists of multiple independent neural networks connected by some intermediary. Each independent neural network, or module, operates on separate inputs and performs some subtask of the overall task. The intermediary, or coordinator, combines the outputs of the modules and produces the final output of the network.

Modular neural networks are mainly used for complex problems that involve multiple subtasks, such as pattern recognition, classification, or optimization. Modular neural networks can have some advantages over single neural networks, such as:

- Faster learning speed: Modular neural networks can learn each subtask separately and in parallel, reducing the training time and complexity.
- Higher accuracy: Modular neural networks can specialize in each subtask and avoid interference or confusion between different inputs or outputs.
- Better generalization: Modular neural networks can adapt to new or unseen data by using the most suitable module or combination of modules.
- Easier maintenance: Modular neural networks can be modified or updated by changing or adding modules without affecting the whole network.

Some of the applications of modular neural networks are:

- Face recognition: Modular neural networks can recognize faces by using different modules for different facial features, such as eyes, nose, mouth, or shape.
- Handwritten digit recognition: Modular neural networks can recognize handwritten digits by using different modules for different strokes, curves, or angles.
- Image segmentation: Modular neural networks can segment images into different regions by using different modules for different colors, textures, or shapes.
- Traveling salesman problem: Modular neural networks can solve the traveling salesman problem by using different modules for different cities, distances, or routes.

These are just some of the examples of modular neural network applications. There are many more possibilities and challenges for modular neural networks in the real world.

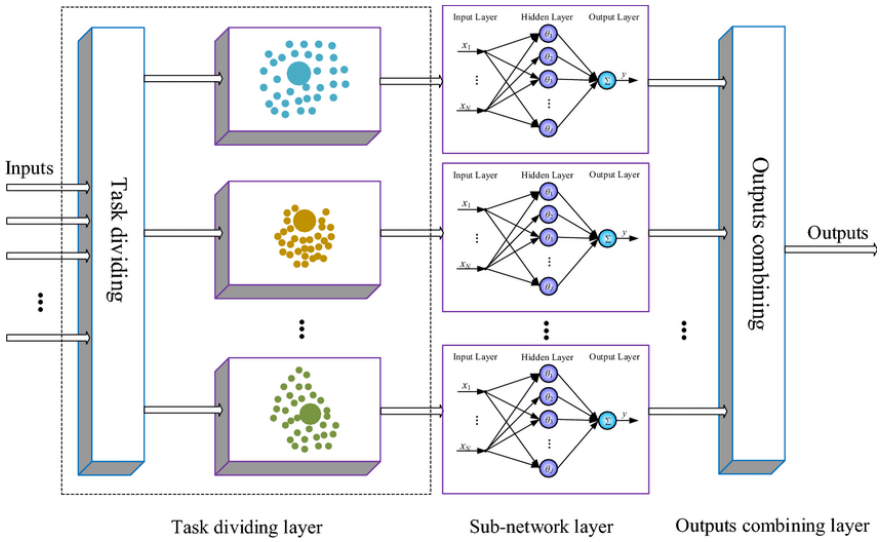

Fig 25: Modular CNN model example (Qiao et al., 2020)

3.4.4.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image classification, object detection and segmentation.

They are inspired by the structure of the human visual cortex and exploit the spatial hierarchies of features learned from training data. CNN uses convolutional layers to scan the input image, applying a filter at each step to produce a condensed version of the image, known as a feature map. These maps are then processed by activation functions and pooling layers to produce the final output. This approach allows CNNs to efficiently handle large amounts of data and maintain spatial relationships between features in the data (Wood, 2021)

CNN uses mathematical operations called convolutions in at least one of its layers, which allows the CNN to detect patterns in images and simplifies at the same time the process to “understand” what an image composition.

This kind of Neural network has three main types of layers: convolutional, pooling, and fully connected. The convolutional layer applies a set of filters to the input data to extract features. The pooling layer reduces the size of the data by applying a function such as max or average. The fully connected layer connects all the nodes from the previous layer to the output layer.

As it was mentioned before, CNN are popular due to their advantage over other approaches in Computer Vision, speech recognition, or audio signal inputs.

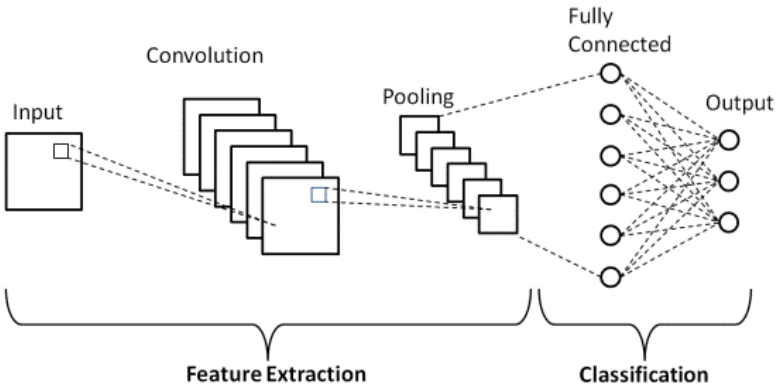

Fig 26: CNN architecture (Balaji, 2020)

3.4.5 Main elements of CNN

A Convolutional Neural Network (CNN) is a type of artificial neural network used for image and video analysis. The main elements of a CNN include:

3.4.5.1 Convolutional layers

Convolutional layers are the layers where filters are applied over the input data, such as images, or to other feature maps in a deep CNN (TowardDataScience, 2019). The filters are weight vectors that cover a patch of the previous layer and extract features by performing a mathematical operation called convolution.

Convolutional layers are the core building blocks of a CNN, and they are where most of the computation occurs. They require a few components, such as input data, a filter, and a feature map. The feature map is the output of the convolutional layer, and it represents the activation of the filter on the input data. (Wood, 2021).

3.4.5.2 Pooling layers

Pooling layers are used to reduce the dimensions of the feature maps generated by the convolutional layers. This helps to reduce the number of parameters to learn, and the amount of computation performed in the network (Khosla, 2023)The pooling layer operates on each feature map separately and creates a new set of feature maps with smaller sizes.

The pooling layer applies a function to a window of values in the feature map, called a pooling window, and outputs a single value that summarizes the values in the window. The most common functions are max pooling, which outputs the maximum value in the window, and average pooling, which outputs the average value in the window.

The size and stride of the pooling window affect the output size of the pooling layer and the degree of dimensionality reduction. A larger window size will result in a smaller output size and a higher degree of dimensionality reduction, but it may also lose some important features. A smaller window size will result in a larger output size and a lower degree of dimensionality reduction, but it may also preserve some important features.

The stride is the number of steps that the pooling window moves across the feature map. A larger stride will result in a smaller output size and a higher degree of dimensionality reduction, but it may also skip some important features. A smaller stride will result in a larger output size and a lower degree of dimensionality reduction, but it may also capture some important features.

There is no definitive rule for choosing the size and stride of the pooling window, but some common choices are 2x2 window with stride 2, which reduces the feature map size by half, or 3x3 window with stride 2, which reduces the feature map size by more than half (Shao et al., 2022)

3.4.5.3 Activation functions

Activation functions are nodes that are added at the end of or in between neural networks. They help to decide if the neuron would fire or not, and they introduce non-linearity to the network, which enables it to learn complex patterns.

There are many types of activation functions, such as linear, polynomial, sigmoid, tanh, ReLU, Leaky ReLU, ELU, SELU, GELU, etc. Each activation function has its own advantages and disadvantages, such as computational efficiency, gradient vanishing or exploding, output range, etc. (Hao et al., 2020)

The choice of activation function depends on the type of problem, the type of layer, and the performance of the network. There is no definitive rule for choosing an activation function, but some common choices are ReLU for hidden layers, which is simple and fast, and SoftMax for output layers, which is suitable for multi-class classification.

Some properties of different activation functions are (Kanth, 2022) :

- Non-linearity: This means that the activation function is not a straight line, and it can capture complex patterns in the data. Most activation functions are non-linear, except for the linear function.
- Continuously differentiable: This means that the activation function has a smooth curve and a well-defined derivative at every point. This is important for gradient-based optimization methods, such as backpropagation. Most activation functions are continuously differentiable, except for ReLU and Leaky ReLU, which have a sharp corner at zero.
- Range: This means that the activation function has a lower and upper bound for its output values. This can help to control the magnitude of the output and prevent gradient exploding or vanishing. Some activation functions have a finite range, such as sigmoid and tanh, which output between 0 and 1, and -1 and 1, respectively. Some activation functions have an infinite range, such as linear, ReLU, Leaky ReLU, ELU, SELU, and GELU, which output any real number.
- Monotonic: This means that the activation function either increases or decreases monotonically as the input increases. This can help to preserve the order of the inputs and outputs. Most activation functions are monotonic, except for tanh and GELU, which have a non-monotonic region around zero.
- Approximates identity near the origin: This means that the activation function behaves like the identity function (f(x) = x) when the input is close to zero. This can help to preserve the information of the input and avoid saturation. Some activation functions approximately identity near the origin, such as tanh, ReLU, Leaky ReLU, ELU, SELU, and GELU. Some activation functions do not approximate identity near the origin, such as linear, sigmoid, and SoftMax.

Here are some mathematical formulas of different activation functions:

- Linear: f(x) = ax + c
- Sigmoid: f(x) = 1 / (1 + e^(-x))
- Tanh: f(x) = (e^(x) - e^(-x)) / (e^(x) + e^(-x))
- ReLU: f(x) = max (0, x)
- Leaky ReLU: f(x) = max (0.01x, x)
- ELU: f(x) = x if x > 0, alpha * (e^(x) - 1) if x <= 0
- SELU: f(x) = lambda * x if x > 0, lambda * alpha * (e^(x) - 1) if x <= 0
- GELU: f(x) = 0.5 * x * (1 + tanh(sqrt(2/pi) * (x + 0.044715 * x^3)))

3.4.5.4 Fully connected layers.

These layers connect every neuron in the previous layer to every neuron in the next layer, allowing the network to make a prediction based on the features extracted by the convolutional and pooling layers.

3.4.5.5 Loss function

This function measures the difference between the predicted output and the ground-truth output and is used to guide the training process by adjusting the weights of the network to minimize the loss.

3.4.5.6 Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. For example, a model trained to recognize cars can be used to initialize a model that recognizes trucks (Donges, 2022).

Transfer learning is useful because it can save time and computational resources by leveraging the knowledge gained from a previous task. This can help to overcome the challenges of insufficient data, domain shift, or complex problems that require a lot of expertise.

Transfer learning is mostly used in deep learning, especially in computer vision and natural language processing, where pre-trained models are widely available and can be fine-tuned or adapted to new tasks. Some examples of transfer learning applications are image classification, object detection, sentiment analysis, text summarization, etc.

These elements work together to form the overall architecture of a CNN and can be combined and configured in various ways to suit the specific requirements of a given problem.

3.4.6 Open Source ready to use tools.

In the previous chapters we talked about algorithms, models, convolutional, and more, but, for face emotion recognition, there are also ready to use tools, such DeepFace and FaceApi, which are libraries for deep facial recognition in Python and JS (FaceApi runs in JS).

In the following section we are going to give a short resume of them.

3.4.6.1 DeepFace

DeepFace is a lightweight face recognition and face attribute analysis library for python (Serengil & Ozpinar, 2020) .

One of the biggest benefits of using a library like DeepFace is its simplicity, this tool includes a complete library with pre-trained models that are considered The State of The Art, which means that they are the most efficient and lightweight models both for detecting faces, compare them as to detect emotions and facial expressions.

DeepFace give access to:

- Face Verification: The task of face verification refers to comparing a face with another to verify if it is a match or not. Hence, face verification is commonly used to compare a candidate's face to another. This can be used to confirm that a physical face matches the one in an ID document.
- Face Recognition: The task refers to finding a face in an image database. Performing face recognition requires running face verification many times.
- Facial Attribute Analysis: The task of facial attribute analysis refers to describing the visual properties of face images. Accordingly, facial attributes analysis is used to extract attributes such as age, gender classification, emotion analysis, or race/ethnicity prediction.
- Real-Time Face Analysis: This feature includes testing face recognition and facial attribute analysis with the real-time video feed of your webcam.

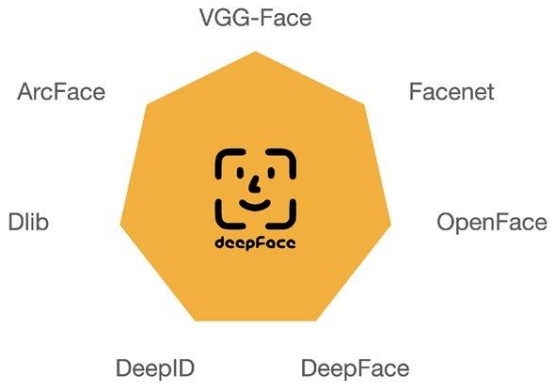

Fig 27: DeepFace Face Detection Methods

3.4.6.2 FaceApi

FaceApi is a JavaScript library that enables developers to use face detection and recognition in their web applications without requiring a background in machine learning.

It is based on the tensorflow.js core API, which allows it to run efficiently and smoothly on the browser. FaceApi can perform various tasks related to face analysis, such as face alignment, landmark detection, emotion recognition, face recognition, age and gender estimation, and face expression recognition.

FaceApi is easy to use and flexible, as it provides multiple models and options for face detection and recognition. It supports four different face detection models: Tiny Face Detector, SSD MobileNet V1, MTCNN, and Tiny YOLOv2. It also supports two different face recognition models: Face Recognition Net and Face Landmark Net. FaceApi allows developers to choose the best model for their use case, depending on the trade-off between speed and accuracy.

FaceApi is an open-source project that is actively maintained and updated by its creator, Vincent Mühler, and other contributors. It has a large and growing community of users and developers who provide feedback, suggestions, and bug reports.

One of the recent updates this tool has come with is the option to run the facial emotion recognition model using its own webpage. This option is useful and practical for new users' first approach for this technology. (Mühler, 2023)

Fig 28: Emotion Recognition with FaceApi example (Mühler, 2023)

3.5 Approaches

The goal of this project is to compare different algorithms and open-source tools for face emotion recognition. Different models with different architecture, number of hidden layers, neurons, image resolution, face cropping and face alignment are going to be used to find the model with the highest accuracy and more simplicity.

The project is going to pursue different approaches for human face emotion recognition, these approaches are:

- Singular Approach
- Semi-singular approach
- Universal approach

3.5.1.1 Methods Selection

As it was mentioned before the goal of this project it is to evaluate open-source tools, based on that we reviewed different available tools and found that the most documented and used tools in the market were DeepFace and FaceApi, both with several YouTube tutorials, active developers constantly improving the libraries and active community using them and open to help others to solve problems.

About the Convolutional Neural Models and architectures, we decided to use TensorFlow because the approach with it was, in the beginning of the research, more user-friendly in comparison with its competitors, like PyTorch.

The used models were the ones available in TensorFlow Hub and Keras (*Keras Applications*, 2023)

When we trained model with lower resolutions (48 x 48) we used those models that allowed us to train them with low resolutions images, models such as ResNet50, does not allow us to train it with images smaller than 96 x 96, the same with others such as SeNet, Xception and ConvNet.

3.5.1.2 Datasets

During this project we have worked with 4 different datasets, 3 of which are for public use (RAF-DB, FER2013 and AffectNet) and one was created specifically for the analysis of the algorithms used with OpenCV (LBPH, Eigenfaces and Fisherfaces).

Dataset Name	Acquisition	Expressions	No. Of Images	Resolution
Real-world affective Database (RAF-DB)	Wild	Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise	~15,339	100 x 100
FER2013 (Facial Expression Recognition 2013 Dataset)	Wild	Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise	~35,887	48 x 48
AffectNet	Wild	Angry, (Contempt), Disgust, Fear, Happy, Neutral, Sad, Surprise	~27,766	512 x 512
Own Dataset	Webcam	Angry, Happy, Sad, Surprise	4800	150 x 150

Fig 29: Datasets used.

3.5.2 Singular approach

The particular approach, as the name mentions, is an approach that cannot be generalized, this is because it's trained using specific individuals' expressions, meaning that, in a normal scenario, with different people, the model won't be able to recognize accurately an estranger's face expression.

In the other hand, will be more accurate to recognize an emotion of the subjects (the 4 individuals)

To train these models we used Eigenfaces, Fisher faces and LBPH methods, which are part of OpenCV's library.

The data gathering was as achieved as follows:

- Using python and a laptop webcam, we capture the individuals faces and with OpenCV we can crop the faces out of the image and save them in the designated folder, this process is repeated with the 4 individuals under the same circumstances (same environment, same light, same place, and time)
- Once the dataset is created, using python, we divide the dataset as the community normally suggests, which is 70/20/10, this is important because is going to help us to not feed the models with images that are going to be used for the accuracy test.
- Once the new dataset is created, we group the images in their correspondence folder, all together, these labels (emotions) are going to be used to test accuracy later, with a simple test consisting of label == expected then is good, if not then is wrong.
- We proceed to train the model, in this approach we can use 3 different models which are integrated in OpenCV library, these 3 models, which were mentioned before are eigenfaces, Fisherfaces and LBPH.

3.5.2.1 Limited to 4 individuals

Once we have trained the models, we are able to see high accuracy but also, as it was mentioned, high limitations, this means that the model will be accurate to detect the emotions if we want to detect them in the 4 individuals that were used to train the model but won't be able to obtain good accuracy with.

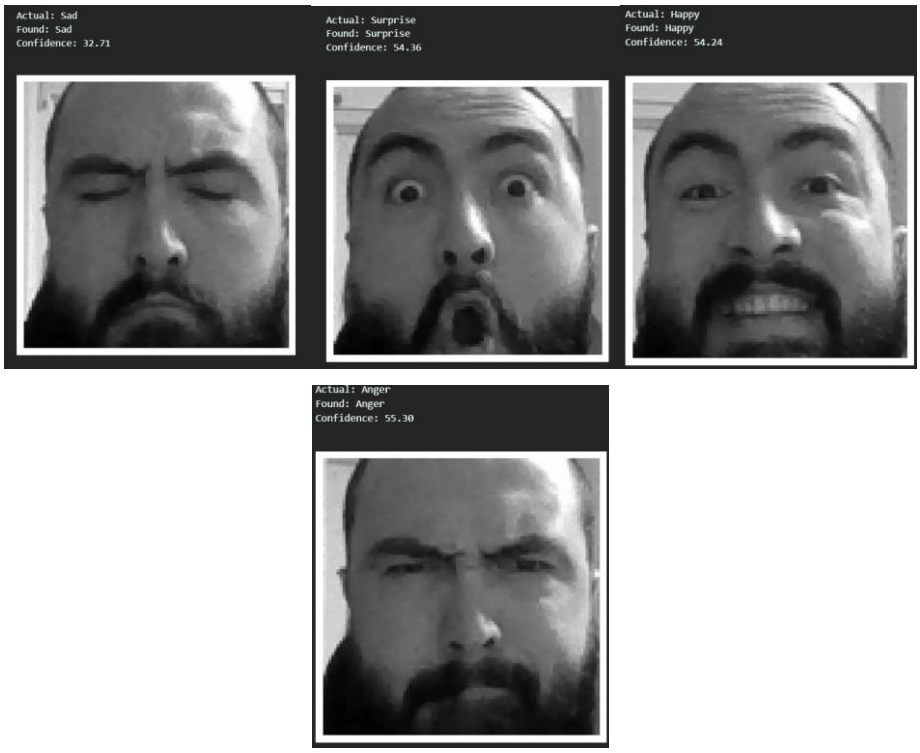

Fig: 30 Particular model images 4 individuals (LBPH)

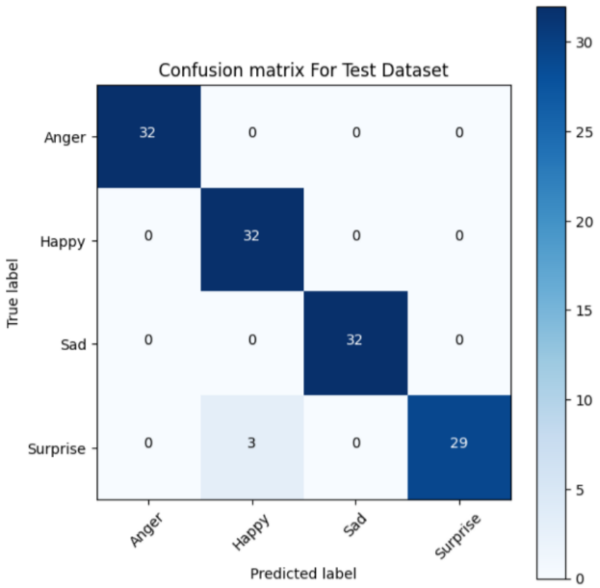

Fig: 31: Particular Model images Confusion Matrix (LBPH)

3.5.2.2 Experimentation with RAF-DB

It's possible as well, to train one of these models with a dataset such as FER2013 or RAF-DB, the accuracy, due to the differences between the faces, will be low and different approaches, like CNN models, might obtain better results.

Fig: 32: Generic approach with LBPH

As we can see, the accuracy of this approach, with a non-particular dataset, is very low.

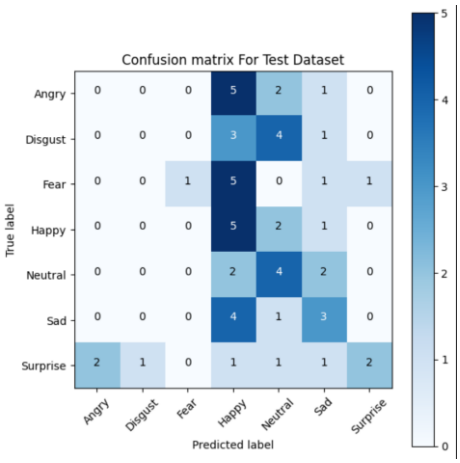

Fig 33: Confusion Matrix Non-Particular Dataset LBPH model

3.5.3 Semi-Singular Approach

This approach can have 2 different approaches, manually and fully-connected-neural model.

With the first approach we identify the points that have significant variations when we show different expressions, for example, when we smile, the points around the border of our mouth, the eyebrows and the cheeks might have a significant variation, knowing that we can proceed to apply the Euclidean distance.

3.5.3.1 Euclidean distance manual calculation

As it was mentioned in the introduction, this approach calculates the variation distance between 2 specific points (in more than one case) and based on that calculation the program discriminates certain emotion, this means that, if the variation in the points around our mouth plus the points around our eyebrows have a variation over 10 (for example) then that means the individual is happy.

This approach is very user friendly and easy to configure, its biggest problem is that it depends on the user distance from the camera, and the expression intensity, this means that being a mathematical calculation, doesn't not include exceptions, and that is a problem that can be solved with a CNN model approach.

Not being able to include exceptions means that we are limited to a range of points variation, if a person does not have the same variation in its facial points, then the program won't be able to detect the emotion, it's a simple approach but limited.

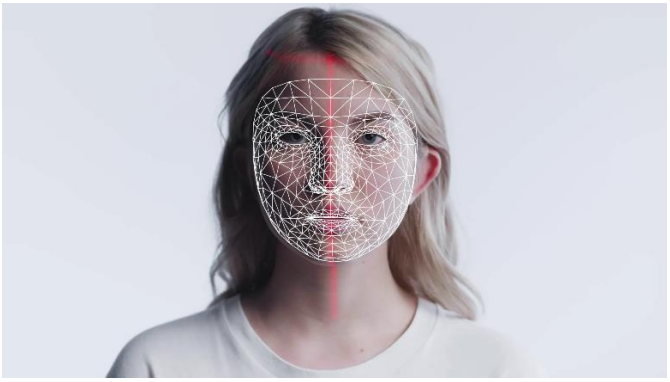

Fig 34. Mediapipe Face Mesh example (Serengil & Ozpinar, 2020)

3.5.4 Fully connected neurons model

This approach is more complex than the Euclidean distance manual calculation, but still not as efficient as a CNN model.

In this approach what we do is extract the face features out of the images (dataset), store them in a csv file and use this csv file with the face features as input in a fully connect model.

A fully connected model is not a CNN model, because it does not apply any convolution, but is powerful enough to find patterns in images (in this case numbers) to come with a possible outcome based on input.

This approach is better than the previous one but still lack of universality accuracy, this means that it might be good with certain emotion which are well represented, but will not be accurate with emotion with lack of representation (data imbalance)

According to the confusion matrix, it has a 68% accuracy but lacks it in those classes with low presence (imbalance)

Classification Report

	precision	recall	f1-score	support
0	0.63	0.44	0.52	66
1	0.43	0.13	0.20	91
2	0.53	0.32	0.40	28
3	0.82	0.85	0.83	412
4	0.59	0.79	0.68	310
5	0.55	0.46	0.50	252
6	0.74	0.79	0.76	202
accuracy			0.68	1361
macro avg	0.61	0.54	0.56	1361
weighted avg	0.66	0.68	0.66	1361

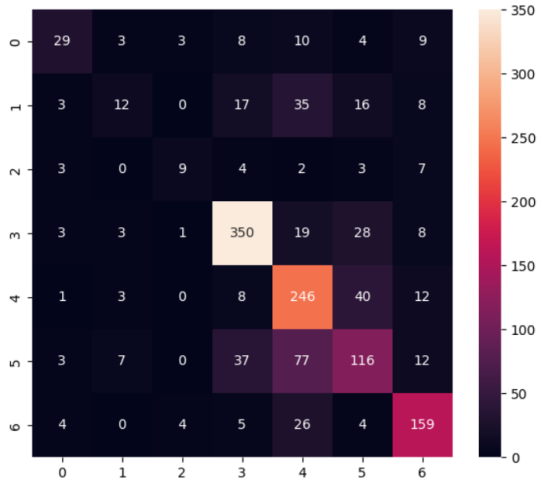

Fig 35: Confusion matrix and Classification report Mediapipe ANN with RAF-DB

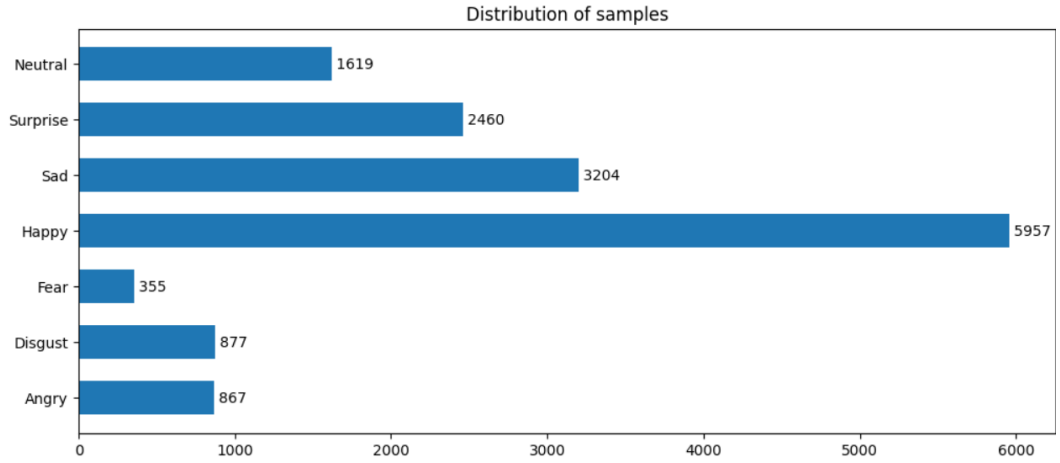

Fig 36: RAF-DB data balance

As we can see, comparing the data balance with the confusion matrix, this model might be more accurate if we feed it with more balance and more complex data, a bigger and more complex dataset, with good quality images and balance weights might give good results and better possibilities for a universalization of the model.

3.5.5 Universal Approach

Universal approach, as the name implies, tries to achieve the emotion detection universally, the goal of this approach is to train a Convolutional Neural Network or to use an open-source library that uses a CNN, so we, with a proper dataset, achieve a face emotion recognition in all the individuals (universal) with the expected limitation of a CNN model (it will hardly get a 100% accuracy).

For this approach we have used Convolutional Neural Networks using TensorFlow and Keras as framework, but also ready to use libraries such as DeepFace and Mediapipe, these 2 libraries already come with their pre-trained models.

For our own CNN models, 3 popular dataset has been used, to test the algorithms and their accuracy (FER2013, RAFDB and AffectNet) and 2 different image resolutions (48 x 48 and 224 x 224), to test the most efficient weight balance approach, we have used 2 different weight balance approaches Oversampling based on Kaggle's number one FER model (Savchenko, 2021) and a Conservative approach, using the same data augmentation steps to increase the model complexity in order to create a more robust model, freezing the same and adding the same layers to maintain the pre-trained weights and additionally, with the Vgg-16 and ResNet50 models, comparing the models performance using 3 different starting points, ImageNet, VGGFace and None pre-trained weight, this approach tried to see if the usage of pre-trained weights with a similar dataset results in a more accurate final result.

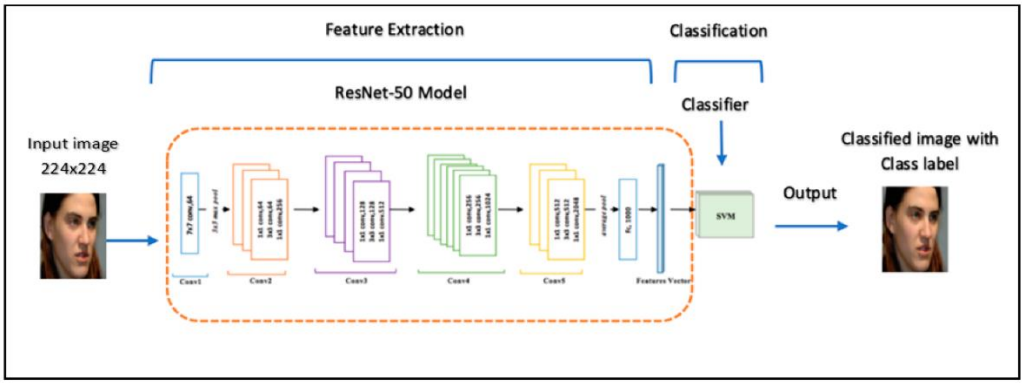

Fig 37: An example of CNN (ResNet50) (Chaurasia, 2020)

3.5.5.1 Experimentation explanation

Being the goal of this project to test the main available models in TensorFlow Keras framework and other open-source python libraries, the idea is to apply the same techniques to all the models:

- freezing the same layers (the top ones),
- add the same new layers (flatten and dense),
- applying the same data augmentation techniques (rotation, sheer, width, and height range, zoom range, horizontal flip, reescalation)
- applying the same pre-process input required for every model.
- using the same pre-trained weights (ImageNet and VGGFace when available)

Here is the test explanation:

Environment and Libraries

1. Deciding for python environment
    a. This depends on the user preferences, regarding AI, the community recommendation is to use Jupyter notebooks, this is because it allows to run the code in parts, helping us to debug possible problems and modify them without running the whole code again.
    b. The required libraries are TensorFlow, Keras layers for the model, matplotlib and NumPy for the data analysis and OpenCV for the real-world testing.

2. Libraries and dependencies
    a. As was mentioned before, we need to import the libraries and needed dependencies to work in python with TensorFlow.
    b. It's important to reduce the computational power, to only import the required libraries, this means that, when we are going to import Keras library, is computational wise to only import those sub libraries we will need, this might help us to speed up the process.

3.5.6 Data Augmentation

We apply transformations to our images, this is important because it will make our model more robust and prepared to detect emotion in faces not only in a face frontal angle, but also with different angles, light conditions and more.

Data augmentation might be tested and refined because too much or too few might be harmful for our models.

3.5.7 Training process

3.5.7.1 Class imbalance and weights

Normally, available datasets are imbalanced, imbalance is when some classes within our dataset are bigger and/or smaller than others, when the differences are not significant, there is not a significant problem either, but when the difference is big then we might face problems, such as overfitting, and we need apply certain strategies to solve it.

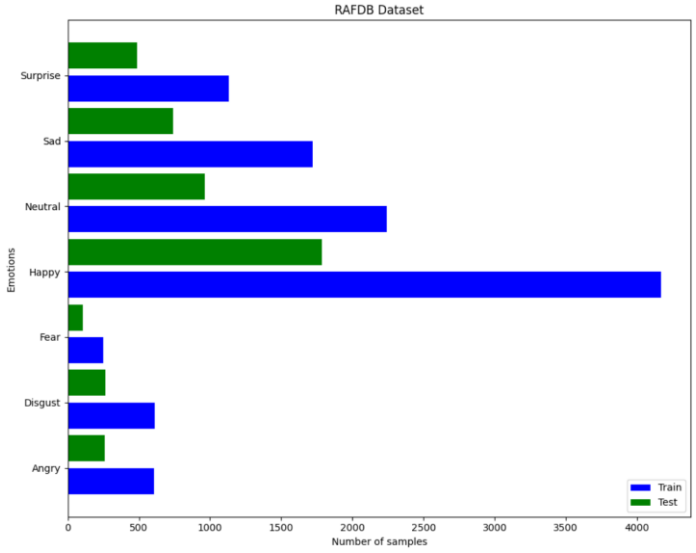

Fig 38: Class balance before weighting

There are different ways to solve this problem, TensorFlow allows us to apply different solutions for imbalance weights, such as manual assigning, oversampling/under sampling, synthetic minority (SMOTE) and other, in our case we applied manual sampling based on the differences between the classes.

3.5.7.2 Oversampling

This part of the code first maps the classes indices and pairs them with the class name, then it computes the frequency of each class in the training data using the “np.unique()” function and stores it in the “counts” variable, it calculates the class weights by dividing the maximum count by the count of each class and stores it in the cw variable. Finally, it normalizes the weights so that the weight of the class with the smallest count is equal to 1 and stores them in the “class_weights” dictionary.

The purpose of computing class weights is to account for class imbalance during training. By assigning higher weights to the minority class and lower weights to the majority class, the model will give more importance to the minority class during training and achieve better performance on it.

```
# Compute the class weights
class_to_idx=validation_generator.class_indices
idx_to_class={class_to_idx[cls]:cls for cls in class_to_idx}
print(idx_to_class)

(unique, counts) = np.unique(train_generator.classes, return_counts=True)
cw=1/counts
cw/=cw.min()
class_weights = {i:cwi for i,cwi in zip(unique,cw)}
print(class_weights)
```

Fig 39: Oversampling Class balance formula (Savchenko, 2021)

This means that we will take the highest value (emotion 3: Happy) as the base number (1) and based on that we will multiply the less represented values as many as times needed to have the same representation as the most represented emotion (happy).

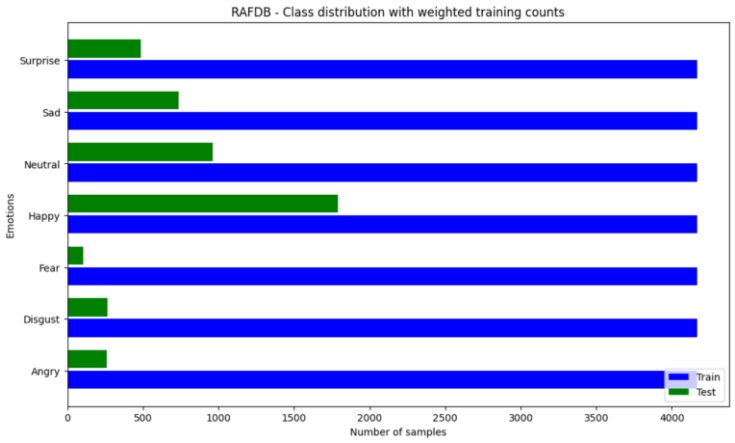

Fig 40: Classes after Savchenko weights application

3.5.7.3 Conservative weight balance

This approach does the same as the previous approach but limiting the augmentation, this is important because it goes against our first idea which is make all the classes even, but at the same time it proves the theory that a dataset with not enough variety might fall into overfitting.

```
counter = Counter(train_generator.classes)
max_val = float(max(counter.values()))
class_weights = {class_id : np.minimum(max_val/num_images,3) for class_id, num_images in counter.items()}
```

Fig. 41: Conservative weight balance

This conservative approach will identify the class with the highest value, in this case Happy and use it as reference, and based on that reference will multiply the other classes values to try to make them even with the main class, here is where the difference with the other approach starts, in this case it will multiply the underrepresented classes but not more than 3 times its value, leaving a class balance like this

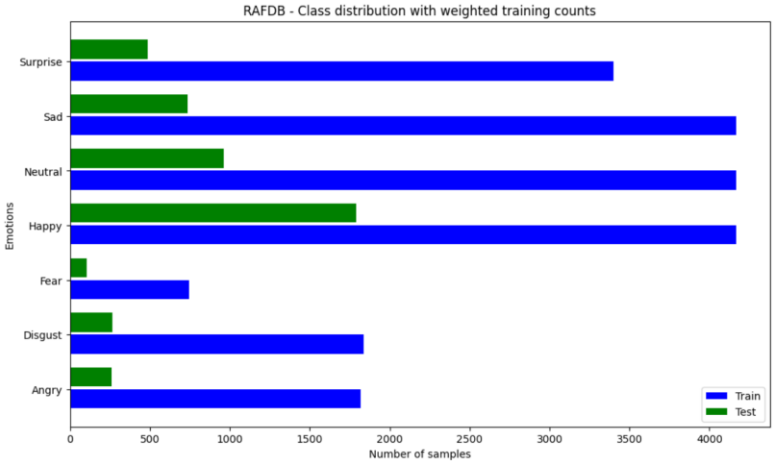

Fig 42: Classes after Conservative balance

These two approaches gave different results under certain circumstances, being the conservative approach better for our datasets, mainly due to the overfitting that the Savchenko approach obtains.

The results are going to be explained in the results chapter.

3.5.8 Callbacks

```
# Checkpoints
filepath="best_model.h5"
checkpoint = ModelCheckpoint(
    filepath,
    monitor='val_accuracy',
    verbose=1,
    save_weights_only=False,
    save_best_only=True,
    mode='max',
    save_freq='epoch')

#Early Stopping
Early = EarlyStopping(verbose=1, patience=20)

#Learning rate reduction
lrd = ReduceLROnPlateau(monitor = 'val_loss',patience = 10, verbose = 1,factor = 0.50, min_lr = 1e-10)
```

Fig 43: Callbacks structure

3.5.8.1 Model Checkpoint

This callback saves the model's best epochs based on the validation accuracy, which is calculated after each epoch during training, so that the best-performing model can be recovered later in case of problems.

The “filepath” is the name that the model is going to be saved after every successful epoch, mode = max, means that the “best” result it’s the one to be saved, and for training purposes, it is better to save not only the weights, but the model itself.

3.5.8.2 Early Stopping

This callback stops the training process early if the model's performance on the validation set does not improve after a certain number of epochs (10), important to avoid overfitting.

3.5.8.3 Reduce LR on Plateau

This callback reduces the learning rate of the model when the validation loss has stopped improving, to help the model converge to a better solution. The monitor argument specifies the metric to monitor, in this case “val_loss”, and the patience argument specifies the number of epochs with no improvement after which the learning rate should be reduced. The factor argument specifies the factor by which the learning rate should be reduced, and the “min_lr” argument specifies the minimum learning rate allowed.

This part of the code allows the model to reduce the learning rate, until 1e-10, which is1 x 10^-10 (1 multiplied by 10 elevated to -10), this allows the model to find the right path for the best possible solution.

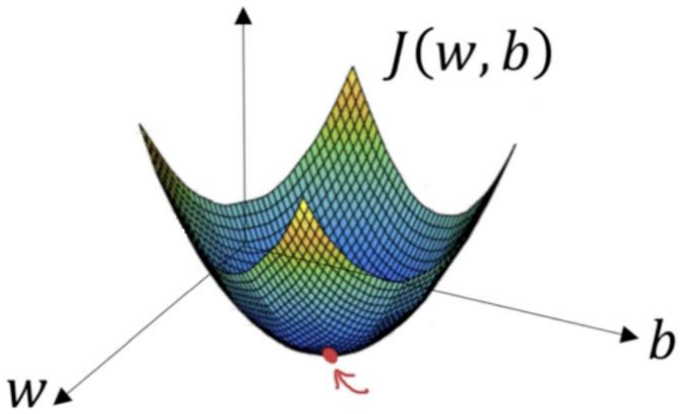

Fig 44: Gradient descent process (Donges & Powers, 2023)

3.5.9 Transfer learning and fine tunning

3.5.9.1 Transfer learning

Transfer learning is a machine learning technique where a pre-trained model is used as a starting point for a new task instead of training a new model from scratch.

The idea is that the pre-trained model has already learned relevant features from a large dataset and can be fine-tuned or adapted to a new dataset for a different but related task (BuiltIn, 2022).

In the context of deep learning, transfer learning typically involves using a pre-trained neural network as a feature extractor by removing the output layer and freezing the weights of the remaining layers. The resulting network can then be used to extract features from new data, which can be fed into a new output layer that is trained specifically for the new task. This approach can significantly reduce the amount of data and computation required for training the new model and can also improve the performance of the model by leveraging the knowledge learned by the pre-trained model.

Transfer learning has been shown to be effective in a variety of applications, including computer vision, natural language processing, and speech recognition. It is particularly useful when the new dataset is small or when training a new model from scratch is computationally expensive or time-consuming.

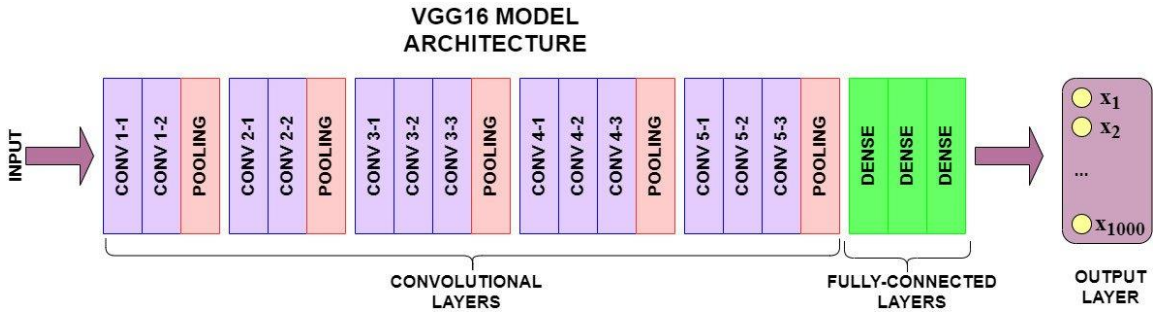

Fig 45: Example transfer learning with VGG16. (Pakhale, 2021)

The idea behind transfer learning is to import a model which has been pre-trained, delete the fully connected layers, which are the one doing the classification and add new layers to work for our purposes, in this case facial emotion recognition.

3.5.9.2 Fine Tunning

Fine-tuning refers to the process of taking a pre-trained neural network model and adapting it to a new, similar task by training it further on new data. In other words, instead of training a model from scratch, we can start with a pre-trained model that has already learned some features and patterns from a large dataset, and then retrain it on a smaller dataset specific to the task we are interested in.

This typically involves freezing the weights of most or all the pre-trained layers and only allowing the weights of a few of the top layers to be updated during training. This approach helps to preserve the useful features learned by the pre-trained model while allowing the top layers to be adjusted to the specific task at hand (DeepLizard, 2023)

The idea, in this project, is to apply transfer learning for some epochs and once the model has adapted to the new dataset, unfreeze it an apply fine tuning to the whole model, this will increase the generalization and, at the same time, will improve the model robustness in real world scenarios.

4 Test Implementation

We have used 3 different datasets (RAF-DB, FER2013 and AffectNet), with 2 different resolutions (48 x 48 and 224 x 224) and 2 different weight balance approaches.

The training process has used RGB images to correspond the pre-trained weights (ImageNet and VGGFace).

The models we have used have been:

48 resolution	
Keras TensorFlow	Own Models
VGG16	AlexNet
VGG19	Ensemble Model
MobileNet	Sequential 3 Conv layer
MobileNetV2	Sequential 4 Conv layer
EfficientNetV2	RepVGG
NasNet Mobile	Sequential Simple Model
	Modular Model
	Single Model Multiple Branches
224 resolution	
VGG16	RepVGG
VGG19	Sequential Simple Model
MobileNet	
MobileNetV2	
EfficientNetV2	
NasNet Mobile	
Xception	
ResNet50	
SeNet50	

Fig 46: Models used with their corresponding resolutions.

We consider these models for emotion recognition, based on previous research for the recognition of facial expressions, such as real-time facial emotion recognition (Kartali et al., 2018) Emotion recognition in still pictures (Datcu & Rothkrantz, 2007) and a Survey for Human Face Expression recognition techniques (Revina & Emmanuel, 2021)

In each case we have frozen the same layers, the top ones, and add the corresponding new layers to match the model requirements and

After the training, every model has been tested with itself (validation accuracy) and with an unseen dataset that has been created using images from google.

5 Test Results and Analysis

To fulfill this project's requirements, we evaluated the open-source tools under these 4 concepts.

- Accuracy

  How accurate is the model to detect emotions in human faces.
- Easiness

  How easy is to make the tool work, which requires less effort and modifications to make it run as expected.
- Practicality

  How useful and adaptable the tool is.
- Technical robustness

  How reliable, secure, and flexible the tool could be.

5.1 Ready Solution

We have evaluated 2 different ready to use tools, these are DeepFace and FaceApi.

5.1.1 DeepFace

DeepFace is a very complete library, it comes with the SoTA models for face detection, and it's relatively simple to use, the library is well explained when we review its repository on GitHub, there are video tutorials available on YouTube, making this option one of the most complete ones and accurate.

The repository is not clear about how the face emotion recognition model has been trained, but it is possible to see that it uses a simple sequential CNN model for 48 x 48 x 1 images (grayscale).

Based on quick training, the model obtains 65% accuracy, which might be considered as the average in this field.

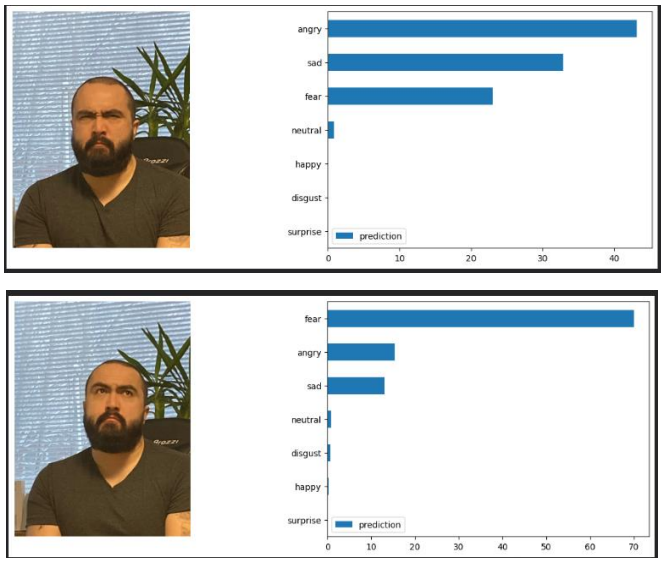

Fig 47: DeepFace Emotion recognition examples

5.1.2 DeepFace Evaluation

Accuracy

The model used in DeepFace is accurate, as always it will depend highly on the input features, but it's possible to say that the model works as expected giving results according to the available tools.

Easiness

The tool is easy to install but requires other libraries to work at their fullest. libraries such as Dlib, VGG-Face and other libraries are not included and it's necessary to install them, besides this issue, the library is easy to use and test.

Practicality

Being able to contain all the SoTA face detection models in one single library, changing between them with just small modifications in the code is practical, making the first approach really welcoming for new users.

Technical Robustness

The library is robust, but could be better, adding more emotion recognition models (architectures) could be useful to test accuracies, combining certain face detection methods and face emotion recognition might give different results and this is something that could be improve.

5.1.3 FaceApi

FaceApi is a lightweight face emotion recognition model that runs using JavaScript, this different approach allows the creator to run the models on browser, making it flexible and easy to use, without any code expertise, the user can test the model just by going to the webpage.

It is possible to run on-site, but this will require installation of some extra modules which differ significantly from the previous ones, like NodeJS.

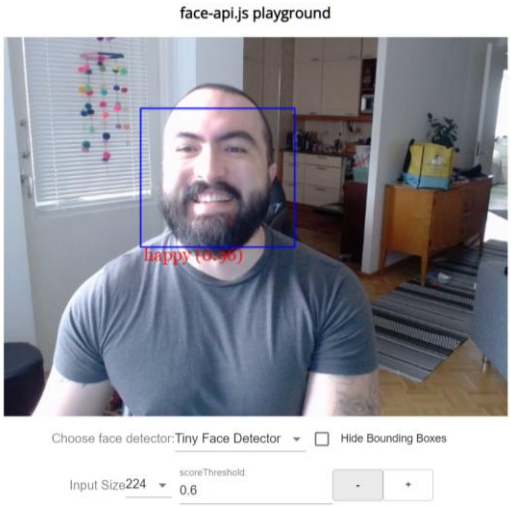

Fig 48: FaceApi example

5.1.4 FaceApi Evaluation

Accuracy

FaceApi uses MobileNet CNN model trained with 224 x 224 images, this means that the accuracy must be around 80%, being consistent with the SoTA.

Easiness

Using and implementing FaceApi is extremely easy, more so if you have experience working with JavaScript.

Practicality

Straight forward model and usage, practical and ready to use.

Technical Robustness

The library is concise, which means that it does exactly what it needs to, its lightweight thanks to mobileNet which allows it to run using browser, it might be good to have more models, but with that we might compromise speed and lightweight.

5.2 Convolutional Neural Networks (CNN)

We have tested different CNN models (Fig 46), all of them using TensorFlow Keras as framework.

In the following tables it is possible to visualize the results both in models trained with a resolution of 48 x 48 (Fig. 49) and with a resolution of 224 x 224 (Fig. 50).

			Savchenko		Conservative	
Model	**Num Parameter**	**Dataset**	**Validation Accuracy**	**Test Accuracy**	**Validation Accuracy**	**Test Accuracy**
		RAF-DB	0,77	0,52	0,79	0,52
VGG16	~15M	FER2013	0,68	0,54	0,68	0,54
		AffectNet 7	0,62	0,54	0,63	0,44
		RAF-DB	0,72	0,38	0,73	0,38
VGG16	~15M	FER2013	0,66	0,48	0,65	0,50
		AffectNet 7	0,60	0,46	0,60	0,45
		RAF-DB	0,73	0,47	0,76	0,51
VGG16	~15M	FER2013	0,67	0,54	0,68	0,52
		AffectNet 7	0,56	0,41	0,57	0,37
		RAF-DB	0,78	0,54	0,78	0,54
VGG19	~20M	FER2013	0,68	0,55	0,67	0,52
		AffectNet 7	0,63	0,44	0,64	0,44
		RAF-DB	0,72	0,42	0,72	0,42
MobileNet	~6M	FER2013	0,67	0,41	0,63	0,41
		AffectNet 7	0,60	0,36	0,62	0,36
		RAF-DB	0,71	0,41	0,65	0,37
MobileNetV2	~5M	FER2013	0,61	0,35	0,62	0,41
		AffectNet 7	0,57	0,46	0,60	0,34
		RAF-DB	0,61	0,35	0,68	0,43
EfficientNetV2_B0	~6M	FER2013	0,59	0,47	0,61	0,43
		AffectNet 7	0,53	0,38	0,50	0,38
		RAF-DB	0,68	0,39	0,67	0,39
NasNet	~4.5M	FER2013	0,60	0,30	0,63	0,29
		AffectNet 7	0,43	0,33	0,50	0,30
		RAF-DB	0,53	0,32	0,63	0,38
AlexNet	~21.5M	FER2013	0,47	0,43	0,59	0,50
		AffectNet 7	0,50	0,43	0,50	0,36
		RAF-DB	0,64	0,40	0,66	0,40
Ensemble	~2M	FER2013	0,61	0,35	0,63	0,41
		AffectNet 7	0,63	0,41	0,63	0,41
		RAF-DB	0,59	0,37	0,72	0,46
Sequential 3Conv	~1.3M	FER2013	0,60	0,49	0,66	0,53
		AffectNet 7	0,60	0,39	0,59	0,41
		RAF-DB	0,65	0,47	0,76	0,54
Sequential 4Conv	~3M	FER2013	0,61	0,52	0,66	0,48
		AffectNet 7	0,59	0,38	0,61	0,40
		RAF-DB	0,63	0,41	0,72	0,45
Modular	~5M	FER2013	0,60	0,49	0,65	0,53
		AffectNet 7	0,56	0,39	0,56	0,38
		RAF-DB	0,60	0,32	0,72	0,46
RepVGG	~26.5M	FER2013	0,65	0,38	0,65	0,20
		AffectNet 7	0,58	0,39	0,58	0,42
		RAF-DB	0,67	0,48	0,75	0,50
Sequential Simple	~2.5M	FER2013	0,61	0,51	0,66	0,55
		AffectNet 7	0,53	0,40	0,46	0,37
		RAF-DB	0,66	0,37	0,69	0,46
SMMB	~5.5M	FER2013	0,52	0,49	0,63	0,51
		AffectNet 7	0,52	0,39	0,46	0,34

Fig 49: 48 x 48 resolution models result.

Model	Num Parameter	Validation Accuracy	Test Accuracy	Validation Accuracy	Test Accuracy
VGG-16 - In	~21M	0,82	0,59	0,83	0,56
VGG-16 - VF	~21M	0,83	0,60	0,85	0,60
VGG-19	~26.5M	0,81	0,59	0,82	0,57
ResNet50 - IN	~49M	0,82	0,56	0,82	0,59
ResNet50 - VF	~24M	0,86	0,62	0,87	0,60
MobileNet	~6M	0,78	0,52	0,81	0,54
MobileNetV2	~5M	0,78	0,53	0,77	0,51
Xception	~46M	0,80	0,54	0,81	0,52
SeNet50 - VF	~27M	0,85	0,59	0,86	0,59
EfficientNet	~6M	0,71	0,57	0,61	0,53
NasNet	~5M	0,75	0,48	0,76	0,49
RepVGG	~26M	0,78	0,56	0,81	0,55
Sequential Simple	~89M	0,61	0,53	0,69	0,47

Fig 50: 224 x 224 resolution models result (RAF-DB).

5.2.1 Pre-Trained weights are important.

It is known that pre-trained weights accelerate the training process and improve the accuracy of the models when the pre-trained weights are like those on the new task.

We have tested the same model 6 times using different approaches related to the pre-trained weights, these are the results.

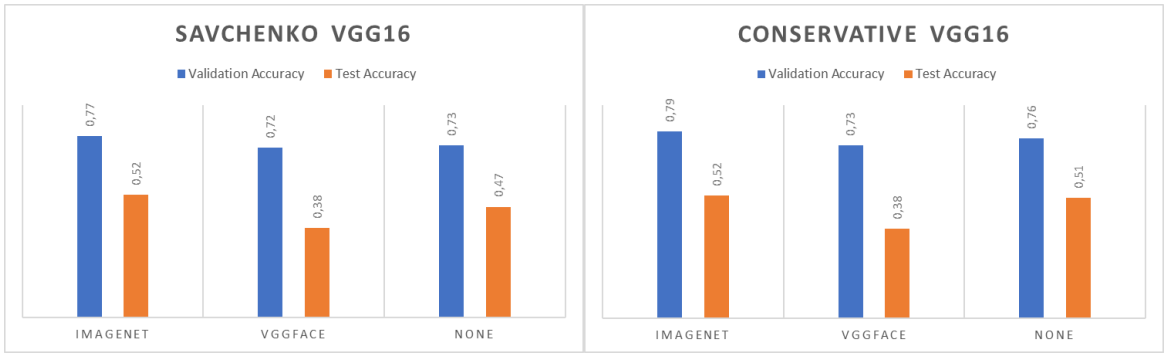

Fig 51: Pre-trained weights importance (48 x 48)

It's possible to see that when we train models using 48 x 48 resolution, using ImageNet as pre-trained weights achieves better results, secondly, we got training a model from scratch (with no pre-trained weights) and less accurate was using VGGFace pre-trained weights.

This might be caused due to the image's resolution, which is something we did not see when training models with bigger resolution.

ImageNet weights were trained using 1000 different classes, while VGGFace used 2622, while ImageNet used different elements as classes, VGGFace only used faces, and with

low resolution, the model is not able to extract the same features as we were able using 224 x 224 later.

5.2.2 Weight balance might affect the model's accuracy.

We found that, due to having imbalance dataset, we should implement a solution to balance the classes, for it we worked with 2 different approaches (as it was mentioned before).

We calculated the average accuracy over all our models and found that applying a conservative weight balance approach (limited by 3) gave us higher accuracy in comparison with a more aggressive weight balance strategy (unlimited).

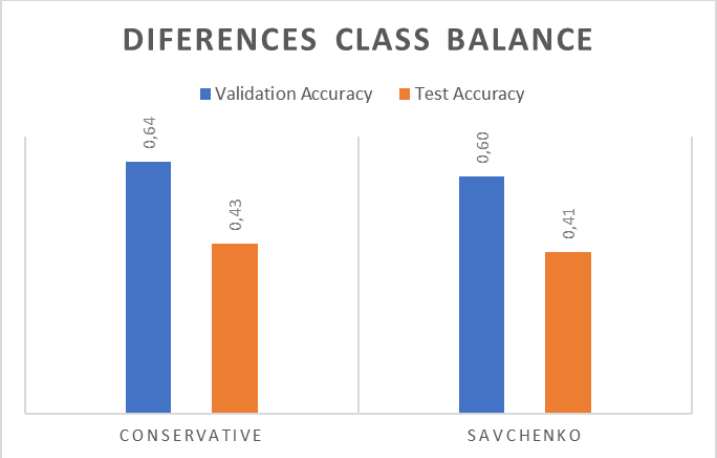

Fig 52: Weight balance differences

We decided to divide the results into pre-trained and none pre-trained models, this to find possible differences between the models, and we also were able to find that, this weight balance differences were more significant with models that did not use pre-trained weights.

The difference was almost 13% using RAF-DB and 10% using FER2013.

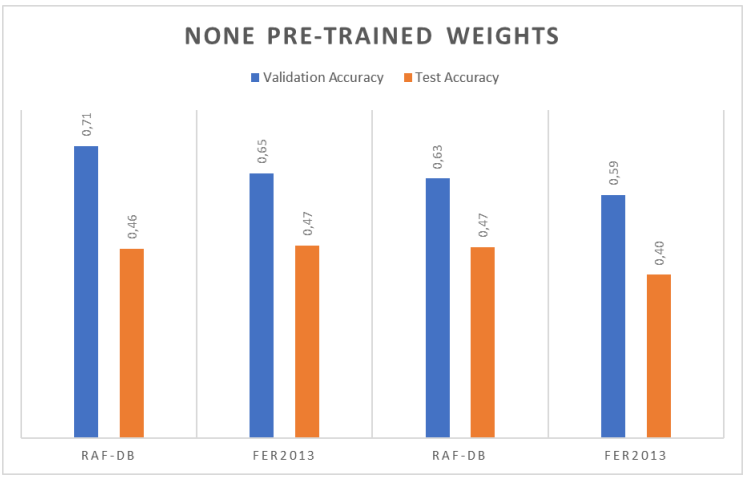

Fig 53: Weight balance differences with NONE pre-trained

This means that, when we train a model from scratch, a conservative approach towards the classes' balances can help our models to improve accuracy and reduce overfitting, this might be related to the pre-trained weights references, since we are training a model from 0.

5.2.3 Importance of a well labeled Dataset

When our goal is to achieve good accuracy, choosing a dataset is important. This was possible to see when we compared the efficiency of all the models trained in 48 x 48 resolution and we compared them using 3 different datasets.

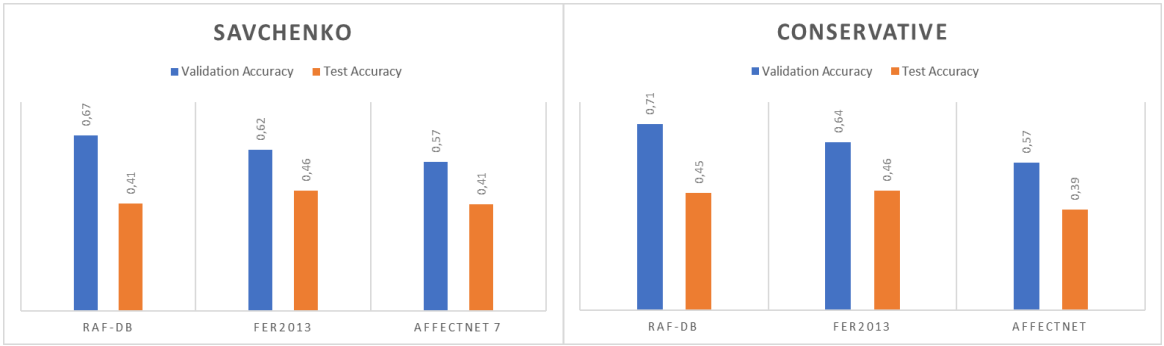

Fig 54: Importance of well labeled dataset

RAF-DB, being a smaller dataset (15k approx.), achieves better accuracy due to its known well labeled strategy, FER2013 (35K) is very consistent with previous results and AffectNet (27K) achieves less accuracy.

What we were able to find is that a smaller dataset might be better than other bigger ones if the quality of the data is better, quality over quantity.

5.2.4 Validation compensates for accuracy.

When we work with limited datasets and low resolutions, and our goal is to achieve good accuracy, we found that is important to have good validation data as well as good quality labeled dataset.

Enough validation data might help us to increase the accuracy of our model, even if the quality of it is not the best.

As it's possible to see in Fig 54, FER2013 gave us a less accurate model, but when we validated it with unseen images, it gave better results than RAF-DB, this might be caused due to FER2013 validation dataset size (7K) against RAF-DB (4K).

5.2.5 More Resolution = More Accuracy

This is a known and very logic topic, the bigger the images the more the algorithm can extract features, this comparison, like the human eye perception, when we see images with more resolution, we are able to detect more details and elements on the image that, working with lower resolutions, we wouldn't be able to.

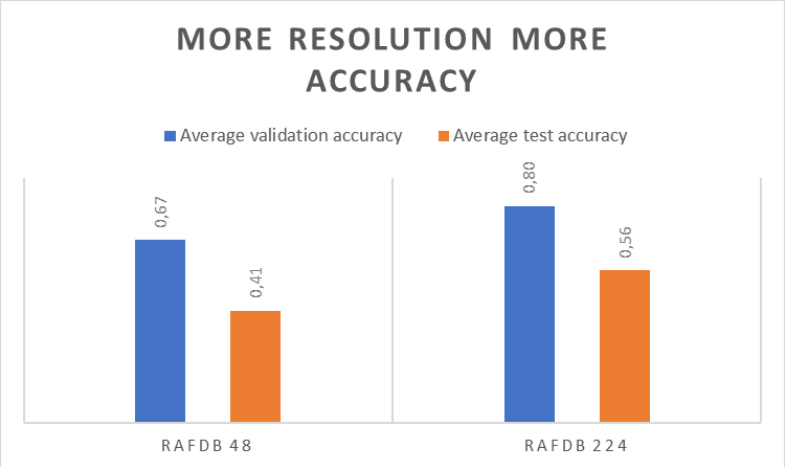

Fig 55: Accuracy comparison with 48 vs 224 resolution

When training a facial emotion recognition model, we need to evaluate if we aim to have fast training process with acceptable accuracy, or we should focus on higher resolution with slower learning speed.

Papers like the impact of Image resolution on Deep Learning (Thambawita et al., 2021), showed that bigger resolutions (512 x 512) obtained 90% in average, using DenseNet and ResNet, while the lowest accuracy (32 x 32) obtained 81%.

In our case, the smaller resolution gave us 67% meanwhile the bigger resolution 0,80 in average, additionally, being this a point to consider when choosing the resolution, the training time per epoch in lower resolution was ~20 secs, while with higher resolution was ~ 115secs, those are aspects we should consider before our training process.

5.2.6 Pre-Trained weights are useful, but Deep CNN models are too.

We understood that training a model using pre-trained weights might increase the model's accuracy in comparison to training a model from scratch.

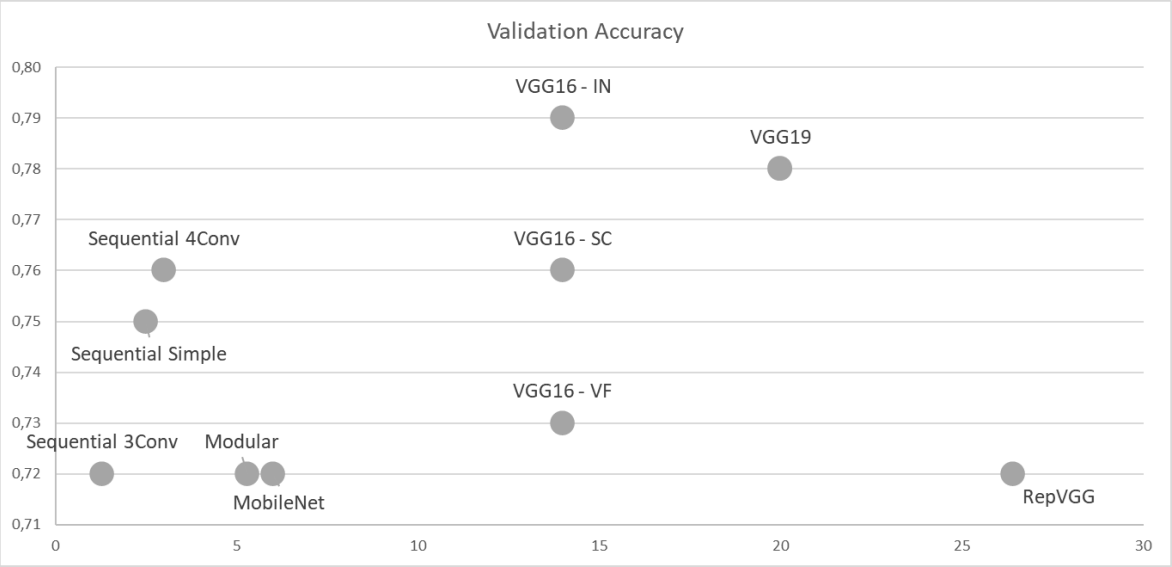

Fig 56: Top best models 48 resolution with parameter numbers

As it is possible to visualize, we got higher accuracy with bigger models such as Vgg19(20M) and Vgg16(15M), but we can get high accuracy also by using smaller models such as MobileNet (5M) or using a VGGNet model with feature extraction such as MediaPipe (7M).

5.2.7 Feature extraction techniques are faster and obtain good results.

A feature extraction method, such as Mediapipe, reads all the images in the dataset and extracts the 468 facial points and stored them in a pickle file, then this file is used as input for the CNN model, giving us a fast-training model (~22 secs) against ~17 sec of Vgg16 but with no pre-trained weights.

5.2.8 Best Models for Facial Expression Recognition

Regarding the CNN models with the best performance, both in 48 resolution and in 224, we can see that the models with the highest precision remain the same.

Models like VGG16, which in low resolution obtained an accuracy of 79% (Fig 49), in high resolution obtained an average of 83% (Fig 50).

In summary, if we want a model with good accuracy, using low resolution, we need to use VGG16 with ImageNet, but if we can sacrifice accuracy for speed, we should focus on a small sequential model with three convolutional layers, Maxpooling, Conv2D and dropout (Fig. 57)

While, if we aim for the highest accuracy, knowing that the training time might be up to 5 times longer, then we should focus on ResNet50 using VGGFace as pre-trained weights, or, for a lighter implementation, MobileNet with ImageNet weights (Fig. 57)

	Model	Num Parameter	Validation Accuracy	Test Accuracy
48	VGG16 - IN	~ 14M	0,79	0,52
	Sequential Simple	~2,5 M	0,75	0,50
224	ResNet50 - VF	~24M	0,87	0,60
	MobileNet	~6M	0,81	0,54

Fig. 57: Best models for FER in 48 and 224 resolutions

5.2.9 CNN Evaluation

Accuracy

The CNN models showed higher accuracy than the open-source ready-to-use tools, not in all cases, but in most of them.

Easiness

The data gathering, the learning process, the research and try and error process while using CNN models it is harder, this means that required more time to learn to use and understand the code for Machine Learning and Computer Vision to obtain good results and achieve SoTA numbers.

Practicality

CNN models are practical, flexible and allow us to do more with the data, it requires more time to comprehend the structure, but it allows us to test different approaches to obtain better results according to our needs, they are

less multi-functional, tools like DeepFace allow us to detect not only emotions but also gender, ethnicity and age, features that are possible to train with a CNN, but they might require also more time

Technical Robustness

CNN models are, maybe, the safest way to manage our data, we can train them using our own hardware which allows us to control where the data goes.

CNN models are also flexible, allow us to train them using different datasets to obtain different results, we can train CNN models to detect ethnicity, age, gender and more.

6 Discussion and Conclusion

Facial Expression Recognition (FER) applications can be seen in a vast variety of fields such as entertainment, medicine, human-computer-interaction (HCI), market research, psychological research and many more.

Regarding our experiments, we used different combinations of dataset, hyper-parameters, resolutions and class balance strategies, our objective was to be able to find the best combination of these in order to achieve the highest possible precision, these different parameters are known and have been used in different investigations and scientific papers, so it is possible to compare its effectiveness in other researches and under different environments.

In this project we were able to verify that the most efficient way to work with FER is the use of CNN models (Fig 49 & 50), We came to this conclusion by comparing the precision of the models trained using algorithms such as Eigenfaces, Fisherfaces and LBPH against convolutional models such as VGG16, MobileNet, among others.

We were able to see that the efficiency and the precision of the convolutional model is superior, not only to detect the expressions with itself, but also to be able to replicate results with new and unseen data, which is the main goal of an efficient CNN model, they should be able to be universally applied.

When working with low resolution images (48 x 48) our results showed that the best approach is by using ImageNet as pre-trained weights, and models like VGG16 and VGG19 or MobileNet and simple sequential model with three convolutional layers (for faster real-world implementation).

When working with higher image resolution (224 x 224) we found similar results related to the best models to pick for the task, adding to them ResNet50, a model very well-known due to its accuracy for image classification purposes, in our case we worked with ResNet50 using VGGFace as pre-trained weights, obtaining 86% accuracy (Fig 57).

Highlight the fact that during this investigation only open source has been used, both for testing and training the models, and for the use of the datasets, for which we have the necessary permissions provided by the creators.

Complying with the new regulations promoted by the EU, the training and development of these models was conducted in controlled environments (sandboxes), following the relevant security measures so as not to compromise the personal information of any individual whose image has been used during the project.

The use of these technologies must be responsible to provide tools to achieve a better coexistence and for no reason should it be used to harm any individual in any sense, as stipulated in the EU guidelines regarding Artificial Intelligence (Yakinova & Ojamo, 2023)

The results obtained in this research can be useful for developers when deciding on open-source tool or CNN model for FER or for similar tasks such as image classification using classes and CNN models.

This project has taught me the importance of well labeled data, which can both improve or worsen the quality of a model based on data augmentation tactics that we can implement.

It was a challenge to work with convolutional networks without previous experience, but it was a task that was made easier thanks to the online community, the available tutorials and the AI boom that has allowed research to be more democratic and available to all.

Regarding the investigation of convolutional networks for the recognition of facial expressions, we must emphasize the intrinsic characteristics of emotions, these are not static moments but a set of constantly changing micro-expressions, for future researches, efforts could be focused on training of models using micro-expressions, this would give greater robustness to the models and could provide greater precision.

References

Anggo, M., & La Arapu. (2018). Face Recognition Using Fisherface Method. *Journal of Physics: Conference Series*, *1028*(1). https://doi.org/10.1088/1742-6596/1028/1/012119

Balaji, S. (2020). *Binary Image classifier CNN using TensorFlow | by Sai Balaji | Techiepedia | Medium*. https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697

Campbell, M., Hoane, A. J., & Hsu, F. H. (2002). Deep Blue. *Artificial Intelligence*, *134*(1–2). https://doi.org/10.1016/S0004-3702(01)00129-1

Chaurasia, A. (2020, March 11). *Exploring ResNets With W&B on Weights & Biases*. https://wandb.ai/site/articles/exploring-resnets-with-w-b

Chihaoui, M., Elkefi, A., Bellil, W., & Amar, C. Ben. (2016). A survey of 2D face recognition techniques. In *Computers* (Vol. 5, Issue 4). https://doi.org/10.3390/computers5040021

Copeland, B. J. (2023, May 30). *Artificial intelligence (AI) | Definition, Examples, Types, Applications, Companies, & Facts | Britannica*. https://www.britannica.com/technology/artificial-intelligence

Datcu, D., & Rothkrantz, L. (2007). Facial expression recognition in still pictures and videos using active appearance models. A comparison approach. *ACM International Conference Proceeding Series*, *285*. https://doi.org/10.1145/1330598.1330717

DeepLizard. (2023). *Fine-tuning a Neural Network explained - deeplizard*. https://deeplizard.com/learn/video/5T-iXNNiwIs

Di Palo, F. (2018). *deep learning - DNN - Is it a good idea to use grayscale images instead of RGB for training - Stack Overflow*. https://stackoverflow.com/questions/53360621/dnn-is-it-a-good-idea-to-use-grayscale-images-instead-of-rgb-for-training

Donges, N. (2022, September 12). *What Is Transfer Learning? A Guide for Deep Learning | Built In*. https://builtin.com/data-science/transfer-learning

Donges, N., & Powers, J. (2023, March 27). *What Is Gradient Descent? | Built In*. https://builtin.com/data-science/gradient-descent

Ekman, P. (1972). Universals and Cultural Differences in Facial Expressions of Emotion BT - Nebraska Symposium on Motivation. *Nebraska Symposium on Motivation*, *19*.

Fenjiro, Y. (2019, July 17). *Face Id: Deep learning for face recognition | by Youssef Fenjiro | Medium*. https://medium.com/@fenjiro/face-id-deep-learning-for-face-recognition-324b50d916d1

Garanhel, M. (2021). *10 top drivers and challenges in computer vision in 2023*. https://www.aiacceleratorinstitute.com/10-top-drivers-and-challenges-in-computer-vision-in-2023/

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11). https://doi.org/10.1145/3422622

Gradilla, R. (2020, July 28). *Multi-task Cascaded Convolutional Networks (MTCNN) for Face Detection and Facial Landmark Alignment | by Rosa Gradilla | Medium*. https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923

Great Learning Team. (2022, March 7). *How to Detect Face Recognition using Viola Jones Algorithm*. https://www.mygreatlearning.com/blog/viola-jones-algorithm/

Gu, L., & Kanade, T. (2008). A generative shape regularization model for robust face alignment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5302 LNCS*(PART 1). https://doi.org/10.1007/978-3-540-88682-2_32

Hao, W., Yizhou, W., Yaqin, L., & Zhili, S. (2020). The Role of Activation Function in CNN. *Proceedings - 2020 2nd International Conference on Information Technology and Computer Application, ITCA 2020*. https://doi.org/10.1109/ITCA52113.2020.00096

IBM. (2022). *What is Computer Vision? | IBM*. https://www.ibm.com/topics/computer-vision

Ilesanmi, A. E., & Ilesanmi, T. O. (2021). Methods for image denoising using convolutional neural network: a review. *Complex and Intelligent Systems*, *7*(5), 2179–2198. https://doi.org/10.1007/s40747-021-00428-4

Jagtap, A. M., Kangale, V., Unune, K., & Gosavi, P. (2019). A study of LBPH, eigenface, fisherface and haar-like features for face recognition using OpenCV. *Proceedings of the International Conference on Intelligent Sustainable Systems, ICISS 2019*. https://doi.org/10.1109/ISS1.2019.8907965

Johnson, J. (2020, July 27). *What's a Deep Neural Network? Deep Nets Explained – BMC Software | Blogs*. https://www.bmc.com/blogs/deep-neural-network/

Kanth, S. (2022, December 2). *What are Activation Functions in Neural Networks?* https://www.mygreatlearning.com/blog/activation-functions/

Kartali, A., Roglic, M., Barjaktarovic, M., Duric-Jovicic, M., & Jankovic, M. M. (2018). Real-time Algorithms for Facial Emotion Recognition: A Comparison of Different Approaches. *2018 14th Symposium on Neural Networks and Applications, NEUREL 2018*. https://doi.org/10.1109/NEUREL.2018.8587011

Keras Applications. (2023). https://keras.io/api/applications/

Khosla, S. (2023). *CNN | Introduction to Pooling Layer - GeeksforGeeks*. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Klinger, N. (2022). *Why Computer Vision Is Difficult? (5 Ways How To Overcome) - viso.ai*. https://viso.ai/computer-vision/why-computer-vision-is-difficult/

Ko, B. C. (2018). A brief review of facial emotion recognition based on visual information. *Sensors (Switzerland)*, *18*(2). https://doi.org/10.3390/s18020401

Krotov, D., & Hopfield, J. J. (2016). Dense associative memory for pattern recognition. *Advances in Neural Information Processing Systems*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11). https://doi.org/10.1109/5.726791

Lotlikar, R. (2022, February). *The history of computer vision - Discussion Forum | Board Infinity*. https://discuss.boardinfinity.com/t/the-history-of-computer-vision/17796

Manmeet Singh. (2020, May 16). *Image Data Augmentation for Facial Recognition | by Manmeet Singh | Medium*. https://manmeet3.medium.com/face-data-augmentation-techniques-ace9e8ddb030

Mühler, V. (2023). *justadudewhohacks/face-api.js: JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js*. FaceApi GitHub Repository. https://github.com/justadudewhohacks/face-api.js/

Nagel, J. L., Stadelmann, P., Ansorge, M., & Pellandini, F. (2002). A low-power VLSI architecture for face verification using elastic graph matching. *European Signal Processing Conference*, *2002-March*.

Nelson, J. (2020, January 26). *Why pre-processing and augmentation matters for computer vision*. https://blog.roboflow.com/why-preprocess-augment/

OpenCV. (2022). *About - OpenCV*. https://opencv.org/about/

Pakhale, T. (2021, June 24). *Transfer Learning using VGG16 in Pytorch | VGG16 Architecture*. https://www.analyticsvidhya.com/blog/2021/06/transfer-learning-using-vgg16-in-pytorch/

Paul Ekman Group. (2022). *Universal Emotions | What are Emotions? | Paul Ekman Group*. https://www.paulekman.com/universal-emotions/

Phuc, L. T. H., Jeon, H. J., Truong, N. T. N., & Hak, J. J. (2019). Applying the haar-cascade algorithm for detecting safety equipment in safety management systems for multipleworking environments. *Electronics (Switzerland)*, *8*(10). https://doi.org/10.3390/electronics8101079

Prakash, J. (2021, June 2). *Non Maximum Suppression: Theory and Implementation in PyTorch*. https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

Pytorch Forum. (2019). *PyTorch RGB_TO_GRAY - PyTorch Forums*. https://discuss.pytorch.org/t/pytorch-rgb-to-gray/50321

Qiao, J. F., Meng, X., Li, W. J., & Wilamowski, B. M. (2020). A novel modular RBF neural network based on a brain-like partition method. *Neural Computing and Applications*, *32*(3). https://doi.org/10.1007/s00521-018-3763-z

RecFaces. (2022). *¿Qué es la visión por computadora? Sistemas de visión artificial y aplicaciones: ejemplos de soluciones — RecFaces*. https://recfaces.com/es/articles/vision-computador-soluciones

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. http://arxiv.org/abs/1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6). https://doi.org/10.1109/TPAMI.2016.2577031

Revina, I. M., & Emmanuel, W. R. S. (2021). A Survey on Human Face Expression Recognition Techniques. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 33, Issue 6). https://doi.org/10.1016/j.jksuci.2018.09.002

Riva, M. (2023, April 14). *Batch Normalization in Convolutional Neural Networks | Baeldung on Computer Science*. https://www.baeldung.com/cs/batch-normalization-cnn

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088). https://doi.org/10.1038/323533a0

Sabottke, C. F., & Spieler, B. M. (2020). The effect of image resolution on deep learning in radiography. *Radiology: Artificial Intelligence*, *2*(1). https://doi.org/10.1148/ryai.2019190015

Sachin. (2023). *Face detection using Cascade Classifier using OpenCV-Python - GeeksforGeeks*. https://www.geeksforgeeks.org/face-detection-using-cascade-classifier-using-opencv-python/

Savchenko, A. V. (2021). Facial expression and attributes recognition based on multi-task learning of lightweight neural networks. *SISY 2021 - IEEE 19th International Symposium on Intelligent Systems and Informatics, Proceedings*. https://doi.org/10.1109/SISY52375.2021.9582508

Saxena, C., & Kourav, P. D. (2014). Noises and Image Denoising Techniques : A Brief Survey. *International Journal of Emerging Technology and Advanced Engineering*, *4*(3).

Serengil, S. I., & Ozpinar, A. (2020). LightFace: A Hybrid Deep Face Recognition Framework. *Proceedings - 2020 Innovations in Intelligent Systems and Applications Conference, ASYU 2020*. https://doi.org/10.1109/ASYU50717.2020.9259802

Shao, Z., Chen, X., Du, L., Chen, L., Du, Y., Zhuang, W., Wei, H., Xie, C., & Wang, Z. (2022). Memory-Efficient CNN Accelerator Based on Interlayer Feature Map Compression. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *69*(2). https://doi.org/10.1109/TCSI.2021.3120312

Sharma, A. (2022, April 11). *Understanding a Real-Time Object Detection Network: You Only Look Once (YOLOv1) - PyImageSearch*. https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S.,

Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587). https://doi.org/10.1038/nature16961

Singh, A. (2019, September 4). *index_9.png (748×482)*. https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/index_9.png

Singh, P. (2021, July 12). *Understanding Face Recognition using LBPH algorithm - Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2021/07/understanding-face-recognition-using-lbph-algorithm/

Skillcate AI. (2022, August 13). *Histogram of Oriented Gradients (HOG) — Simplest Intuition | by Skillcate AI | Medium*. https://medium.com/@skillcate/histogram-of-oriented-gradients-hog-simplest-intuition-2392995f8010

Thambawita, V., Strümke, I., Hicks, S. A., Halvorsen, P., Parasa, S., & Riegler, M. A. (2021). Impact of image resolution on deep learning performance in endoscopy image classification: An experimental study using a large dataset of endoscopic images. *Diagnostics*, *11*(12). https://doi.org/10.3390/diagnostics11122183

Turing, A. M. (2012). Computing machinery and intelligence. In *Machine Intelligence: Perspectives on the Computational Model*. https://doi.org/10.1525/9780520318267-013

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*(1). https://doi.org/10.1162/jocn.1991.3.1.71

Vemou, K., & Horvath, A. (2021). *Facial Emotion Recognition*. Facial Emotion Recognition. https://edps.europa.eu/system/files/2021-05/21-05-26_techdispatch-facial-emotion-recognition_ref_en.pdf

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). *Pointer Networks*. http://arxiv.org/abs/1506.03134

Viola, P., & Jones, M. J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, *57*(2). https://doi.org/10.1023/B:VISI.0000013087.49260.fb

Wang, W., Xu, K., Niu, H., & Miao, X. (2020). Emotion Recognition of Students Based on Facial Expressions in Online Education Based on the Perspective of Computer Simulation. *Complexity*, *2020*. https://doi.org/10.1155/2020/4065207

What is Artificial Intelligence (AI) ? | IBM. (2023). https://www.ibm.com/topics/artificial-intelligence

Wood, T. (2021). *Convolutional Neural Network Definition | DeepAI*. https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

Yakinova, Y., & Ojamo, J. (2023, May 11). *AI Act: a step closer to the first rules on Artificial Intelligence | News | European Parliament*. https://www.europarl.europa.eu/news/en/press-room/20230505IPR84904/ai-act-a-step-closer-to-the-first-rules-on-artificial-intelligence

Yan, J., Zhang, X., Lei, Z., & Li, S. Z. (2014). Face detection by structural models. *Image and Vision Computing*, *32*(10), 790–799. https://doi.org/10.1016/j.imavis.2013.12.004

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, *23*(10). https://doi.org/10.1109/LSP.2016.2603342

Zhou, B., Lapedriza, A., Torralba, A., & Oliva, A. (2017). Places: An Image Database for Deep Scene Understanding. *Journal of Vision*, *17*(10). https://doi.org/10.1167/17.10.296