NOVEMBER 2023 M.Sc. in Electrical-Electronics Engineering DHUFR AL-OBAIDI

REPUBLIC OF TÜRKİYE

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

PERSONALITY PREDICTION SYSTEM BASED ON PHYSIOGNOMY USING FACE RECOGNITION

M.Sc. THESIS

IN

ELECTRICAL-ELECTRONICS ENGINEERING

BY

DHUFR AL-OBAIDI

NOVEMBER 2023

PERSONALITY PREDICTION SYSTEM BASED ON PHYSIOGNOMY USING FACE RECOGNITION

M.Sc. Thesis

in

Electrical-Electronics Engineering

Gaziantep University

Supervisor

Asst. Prof. Dr. Seydi KAÇMAZ

By

Dhufr AL-OBAIDI

November 2023

PERSONALITY PREDICTION SYSTEM BASED ON PHYSIOGNOMY USING FACE RECOGNITION

submitted by **Dhufr AL-OBAIDI** in partial fulfillment of the requirements for the degree of Master of Science in **Electrical-Electronics Engineering, Gaziantep University** is approved by,

Prof. Dr. Çiğdem AYKAÇ
Director of the Graduate School of Natural and Applied Sciences. …….....

Prof. Dr. Ergun ERÇELEBİ
Head of the Department of Electrical and Electronics Engineering …….....

Asst. Prof. Dr Seydi KAÇMAZ
Supervisor, Electrical and Electronics Engineering
Gaziantep University …….....

Exam Date: 17 November 2023

Examining Committee Members:

Asst. Prof. Dr. Seydi KAÇMAZ
Gaziantep University …….....

Asst. Prof. Dr.Ali ARSLAN
Gaziantep University …….....

Asst. Prof. Dr. Mehmet ARICI
Gaziantep Islam, Science and Technology …….....

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Dhufr AL-OBAIDI

ABSTRACT

PERSONALITY PREDICTION SYSTEM BASED ON PHYSIOGNOMY USING FACE RECOGNITION

AL-OBAIDI, Dhufr
M.Sc. in Electrical-Electronics Engineering
Supervisor Asst. Prof. Dr. Seydi KAÇMAZ
November 2023

97 pages

the human face is a wealthy source of information, with distinct features such as the eyes, nose, and mouth offering a wealth of data. Facial feature recognition algorithms leverage this wealth of information to distinguish one person from another, making it a powerful tool with many applications. This technology has found its way into numerous aspects of our daily lives, from unlocking smartphones with a simple glance to enhancing security in public spaces. Even more, a person's personality traits can be predicted from their outer appearance according to physiognomy as each shape of facial features can tell a lot about a person's personality; this work addresses this by proposing a novel dataset created for facial feature recognition based on their shape and color, specially designed for the YOLO object detection models; the final dataset contains 2,116 images with Over 10K annotations with six classes, namely blue eye, brown eye, rounded nose, rounded eyebrows, pointy nose, and straight eyebrows, employed in all version of the YOLOv8 object detection model, experiment result shows that the small version of YOLOv8 performed the best based on mAP of 89.9%, this work also proposed a modified lightweight version of YOLOv8 with only 211 layers which outperformed the best model by 1.4% as it reaches 91.3%mAP on the proposed dataset.

Key Words: physiognomy, deep learning, computer vision, object detection, YOLO

ÖZET

YÜZ TANIMA KULLANILARAK FİZYOGNOMİYE DAYALI KİŞİLİK TAHMİN SİSTEMİ

AL-OBAIDI, Dhufr
Yüksek Lisans Tezi, Elektrik ve Elektronik Mühendislik
Danışman: Dr. Öğr. Üyesi Seydi KAÇMAZ
Kasım 2023
97 sayfa

İnsan yüzü, gözler, burun ve ağız gibi belirgin özelliklerle dolu bilgi kaynağıdır ve bu bilgi bolluğunu kullanarak yüz tanıma algoritmaları bir kişiyi diğerinden ayırt etmek için bu bilgiyi kullanır; bu, birçok uygulamaya sahip güçlü bir araç haline gelir. Bu teknoloji, günlük yaşamımızın birçok yönüne girmiştir, basit bir bakışla akıllı telefonları açmaktan kamu alanlarında güvenliği artırmaya kadar. Dahası, bir kişinin kişilik özellikleri fizyonomiye göre dış görünüşlerinden tahmin edilebilir; her yüz özelliği şekli bir kişinin kişiliği hakkında çok şey anlatabilir; bu çalışma, YOLO nesne tespit modelleri için özel olarak tasarlanmış olan şekil ve renklerine göre yüz özelliklerinin tanınması için oluşturulan yeni bir veri kümesini önererek bu konuya ele alır; son veri kümesi, YOLOv8 nesne tespit modelinin tüm sürümlerinde kullanılan mavi göz, kahverengi göz, yuvarlak burun, yuvarlak kaş, sivri burun ve düz kaş olmak üzere altı sınıf içeren 2.116 görüntü ve 10.000'den fazla etiket içerir; deney sonuçları, YOLOv8'in küçük sürümünün 89.9% mAP'ye dayalı olarak en iyi performansı gösterdiğini göstermektedir; bu çalışma ayrıca yalnızca 211 katmana sahip modifiye edilmiş hafif bir YOLOv8 sürümünü önerdi ve önerilen veri kümesinde %91.3 mAP'ye ulaştığı için en iyi modeli %1.4'lük bir farkla aştı.

Anahtar Kelimeler: fizyonomi, derin öğrenme, bilgisayarlı görme, nesne algılama, YOLO

‘’ To those I lost during this journey, my father and Elias, may God bless your pure souls; your memory will always remain in my heart.

To my mother, who sacrificed her time and health, I wouldn't be here without you.

To my sister and brothers, without you, nothing in my life would be complete.’’

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to the following individuals, whose unwavering support and encouragement have been instrumental in the completion of this thesis:

First and foremost, I extend my deepest appreciation to my dedicated supervisor, **Asst.** Prof. Dr. Seydi KAÇMAZ for his invaluable guidance, mentorship, and unwavering belief in my abilities. His expertise and constant encouragement have been pivotal in shaping this research.

I am profoundly grateful to my mother, whose enduring love and unwavering belief in me have been a constant source of strength throughout this journey. Her sacrifices and encouragement have been the bedrock of my academic pursuits.

To my wonderful sister and brothers, your unwavering support, understanding, and occasional distractions have kept me grounded and motivated. Your belief in me has been my driving force.

I must acknowledge the incredible friends who stood by me during the long hours of research, providing both moral and emotional support.

To all these remarkable individuals, I owe the successful completion of this thesis. Your contributions, whether big or small, have left an indelible mark on my academic journey. Thank you for being an integral part of this significant achievement.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

ML	Machine learning
DL	Deep learning
ANN	Artificial neural network
DNN	Deep neural network
GPU	Graphical processing unit
XOR	Exclusive OR
AI	Artificial intelligent
ReLU	Rectified linear unit
Tanh	Hyperbolic Tangent
CNN	Convolutional neural network
YOLO	You only look once
SSD	Single shot detector
RPN	Region Proposal Network
NFL	Nearest feature line
FL	feature line
FPGA	Field programmable gate array
AFLW	Annotated Facial Landmarks in the Wild
LFW	Labeled Faces in the Wild
FDDB	Function Designator Data Base
VGA	Video Graphics Array
SVM	Support vector machine
IQ	Intelligence quotient
mAP	Mean average precision
GUI	Graphical user interface
FFHQ	Flicker face high quality
PNG	Portable Network Graphic
TFRecords	Tensor flow records
COCO	Common Objects in Context

JPG	Joint Photographic Experts Group
SiLU	Sigmoid Linear Unit
SPPF	Fast Spatial Pyramid Pooling
C2f	coarse-to-fine
IoU	Intersection over Union
CIOU	Complete intersection over union
PR	precision/recall
AP	Average Precision
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
SGD	Stochastic Gradient Descent
PyQt	Python Qt
Qt	Qtoolkit

CHAPTER 1

INTRODUCTION

1.1 Physiognomy

The practice of detaching a person's character or personality from their outer appearance, especially the face, is known as Physiognomy. Since ancient times, being able to peek a person's personality to their face idea dates back to the ancient Chinese, Egyptian, and Greek.

People have tried to establish a connection between facial morphological features and individual personality traits [1]. Faces play a major part in people's everyday perceptions of other people, according to psychological studies [2]. Unconsciously, humans make trait assessments based on their faces, which could have a significant impact on the outcomes of crucial social events such as elections [3] and court sentences. [4]

It was popularized by the Swiss poet Johann Lavater in the late 18th century, whose ideas became a point of discussion in intellectual circles. Since in Darwin's day, they were more or less taken for granted. It was only after the subject became associated with phrenology, which fell into disrepute in the late 19th century, that Physiognomy was repealed as a pseudoscience. Nowadays, we are witnessing a revival of this science as researchers around the world are re-evaluating what we see in the face, investigating whether the face can give us a glimpse or can reveal the secrets of a person's personality, as some research shows that we make a 100ms snap judgment of others. [5] What is now emerging is a more accurate "new physiological science," But it's no less awesome than its old incarnation. [6].

1.2 Computer Vision

Computer vision is one of the most advanced artificial intelligence technologies that simulate the human’s vision system, attempting to make the computer see objects as the human sees them. The data we generate daily are essential for building a computer vision system as they are used to train the system. In addition, the big difficulty is the need for large computing power necessary for training and testing the system before

deploying it was a problem in the past; even supercomputers used to take days to complete all the required calculations. But, nowadays, with the recent advanced and fast computers, it is possible to analyze and process a large amount of data within hours or even minutes. Computer vision made its first commercial use in the early 1970s, as it was used for line labeling and other applications. Computer vision abilities were limited in their capacity, and the tasks required lots of coding and manually extract of features, but with the advancements in Deep Learning models and neural networks, the field took a giant leap in a way that it was able to exceed humans' potentials in some objects detecting and labeling tasks. Less coding is needed with an automatic feature extraction ability that makes it easier to design applications. Modern Computer vision technology is associated with learning algorithms depending on pattern recognition, so in order for a computer to recognize these patterns, it must be fed with a large amount of data, maybe thousands or even hundreds of thousands of classified images, and then pass it throw the certain algorithm such as convolutional neural network (CNN) which allow the computer to locate these pattern in which each pattern associated with a certain class [7].

1.3 Machine Learning

"Machine learning is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment."[8].

The main idea of machine learning is to make a computer solve a problem without the need to program it. This idea was first presented by "Arthur Samuel" in 1959 when he introduced the first self-learning checkers game, and the term "Machine Learning" was brought to the world by him [9].

In terms of computer vision, Machine learning revealed a different way of problem-solving. In which the computer can recognize patterns as features rather than having programmers manually code each rule into the vision system. After that, classify the images and identify the objects using a statistical technique like support vector machine algorithms or decision trees to find specific patterns [10].

According to the type of data labeling, machine learning algorithms can be classified as supervised, unsupervised, or semi-supervised, as shown in Figure 1.1. An unknown (input, output) mapping is estimated using supervised learning from samples of known.

(Input and output), where the output is labeled (classification and regression).

In unsupervised learning, the learning system only provides samples (such as

clustering and probability density function). When a piece of the data is partially labeled, and the labeled portion is used to infer the unlabeled portion, this process is known as semi-supervised learning (e.g., text/image retrieval systems) [8].

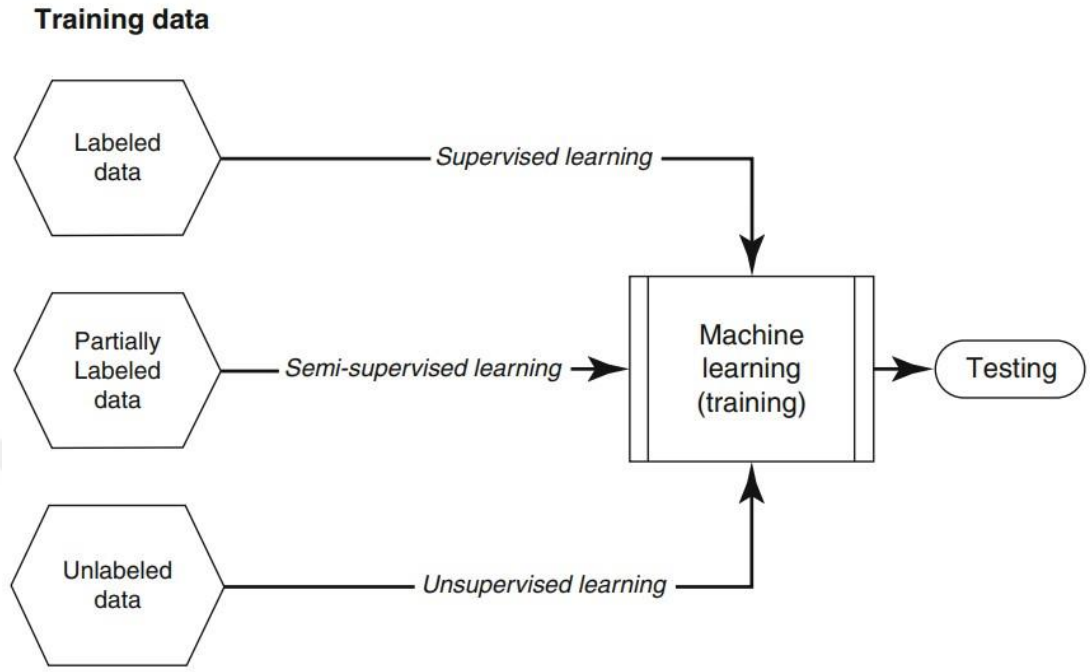

Figure 1.1. Categories of ML algorithms according to training data nature [8].

1.3.1 Deep Learning

As shown in Figure 1.2. Deep learning (DL) is a refined Artificial Intelligence (AI) application that derives from machine learning (ML), but it offers a distinct learning methodology that depends on artificial neural networks (ANNs) to learn and solve problems. ML first heard the phrases "DL" and "ANN" in 1986 [11] and 2000 [12], respectively.

Unlike ML, where features are utilized to build a model that categorizes the image's objects, DL automatically extracts the features from images.

In DL, a task, such as categorization, is assigned to an ANN in addition to raw data, and the ANN learns how to learn to do it automatically [13].

A deep neural network (DNN), also known as a multi-layered ANN, is used to process the input and discover its representation. By using the backpropagation process to demonstrate how a model should change its parameters, which are utilized to generate the representation between layers, DL may find complicated structures within massive

datasets. In contrast to DL, which improves as the size of the data increases, the performance of ML models plateaus once a certain number of examples are added [14] Thanks to today's fast graphical processing unit (GPU), DL has surpassed machine learning abilities and overtaken the computer vision world; application like face recognition, auto parking, autonomous-driving cancer detection, text generation, and

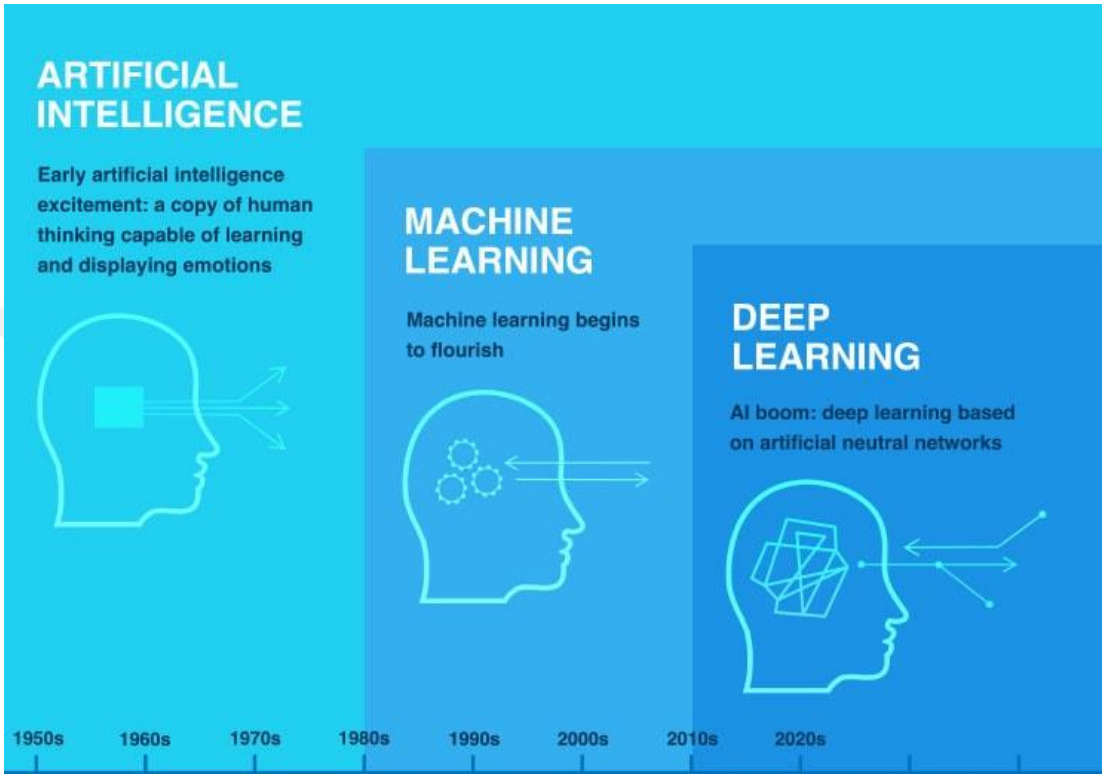

Figure 1.2. Deep Learning (DL) derivation. [15]

voice recognition was made possible, the fact that DNN model can automate these process even with a large amount of data, By incorporating representations that be expressed in terms of other, simpler representations, deep learning addresses this fundamental issue in representation learning. The machine can create complicated notions from simpler ones through deep learning.

Figure 1.3.Demonstrates how a deep learning system can combine basic concepts, such as corners and contours, which are in turn described in terms of edges, to represent the concept of an image of a person [15].

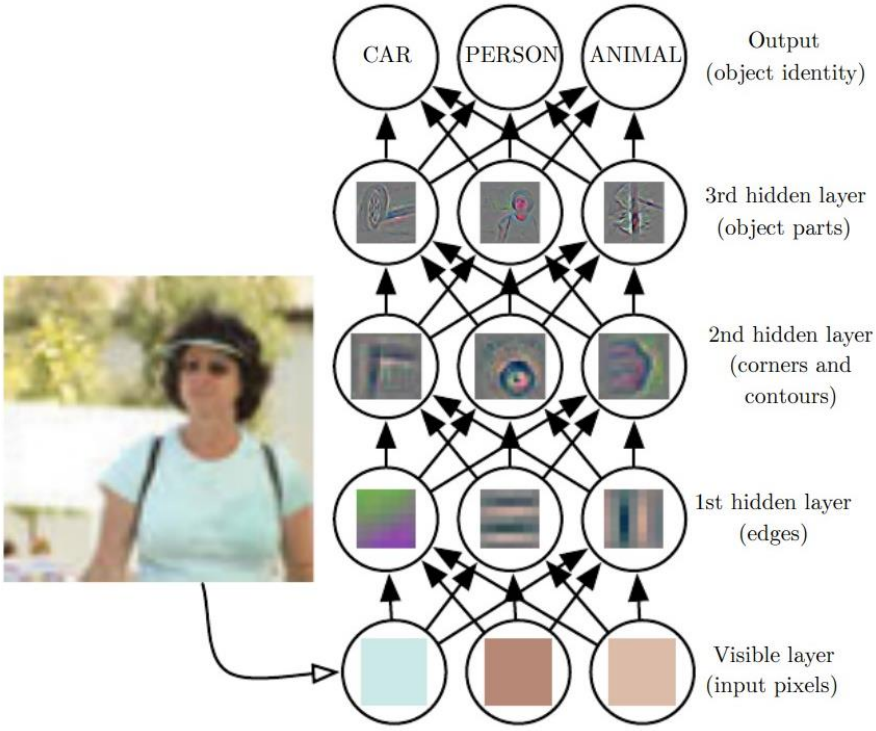

Figure 1.2. How deep learning works. [16]

From the computer perspective, the image in Figure 1.3 it's nothing but a raw sensory collection of pixel value; therefore, it's very challenging for the computer to comprehend those input data, and it takes a lot of work to identify and map a set of pixels to an object identity.

This problem is solved by deep learning, which divides the required complex mapping into a number of nested simple mappings, each of which is specified by a separate layer of the model. The visible layer, so named because it includes the variables we can see, is where the input is presented. The image's features are then gradually extracted via a series of hidden layers[16].

1.4 Artificial Neural Networks

In biology, a neural network is a collection of neurons that are linked together to form a neurological system in both humans and animals[17].

Artificial neural networks (ANNs) are computer programs that mimic how biological neurons process information. ANNs learn (or are trained) through experience rather than programming, and they learn by identifying patterns and relationships in data and turning it into a mathematical model that would help distinguish future data [18].

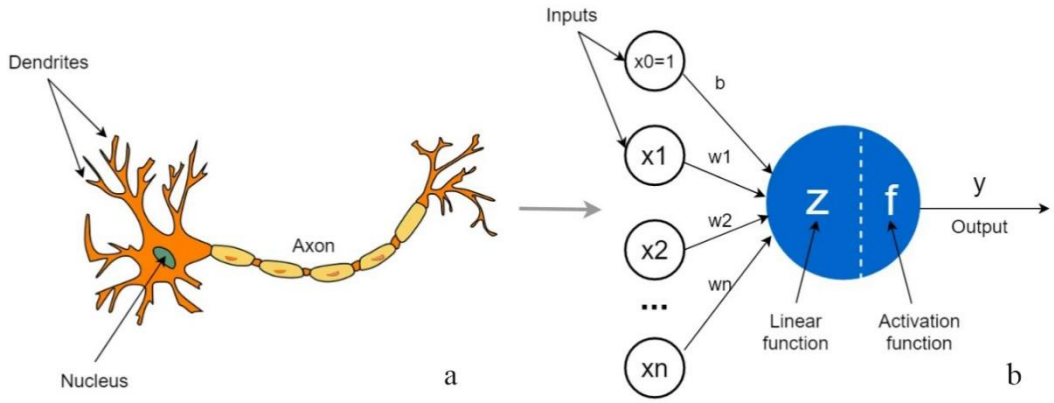

Figure 1.3. both biological and artificial neurons (a,b, respectively [17]

The first mathematical representation of a neural network was presented in 1943 by "Warren McCulloch" and "Walter Pitts" [19], but in 1969, "Marvin Minsky" and "Seymour Papert" conducted research that revealed significant flaws in the perceptron and came to the conclusion that a perceptron is not capable of learning any complex logic. In fact, they demonstrated that even understanding basic logical operations like XOR would be challenging. This resulted in a decline in interest in machine learning generally and neural networks specifically, and it also marked the beginning of the so-called "AI winter." AI should have been taken more seriously by researchers since they believed it was unable to handle difficult challenges. The limitations of the hardware capabilities at the time were one of the main causes of the so-called AI winter. It was impossible to obtain the required computer power, or it was too expensive. The AI winter began to break toward the end of the 1990s as a result of developments in distributed computing that made infrastructure freely accessible and reasonably priced. The thaw rekindled interest in AI research. Eventually, this transformed the present era into an era that might be referred to as the "AI spring," in which there is a great

deal of interest in AI in general and neural networks in particular[20].

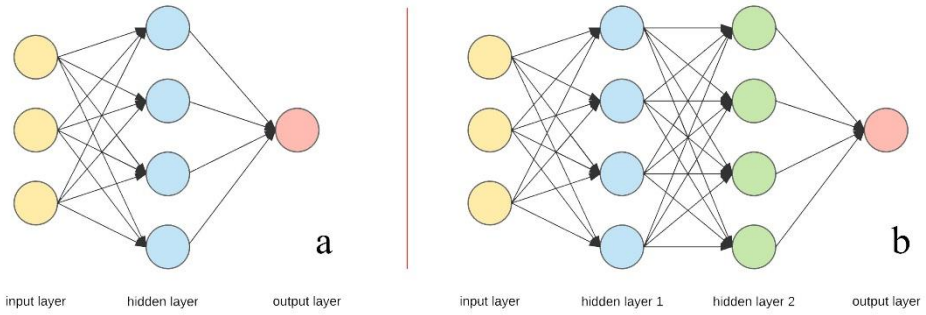

Figure 1.4. Simple neural network (a), multi-layer perceptron (b).

As shown in Figure 1.5. (a), a standard ANN consists of an input layer, hidden layers, and output layer. A multi-layer perceptron is a network with several hidden layers that uses the backpropagation algorithm, as shown in Figure 1.5. (b). [21]

According to Figure (1.4.b), a single artificial neuron consists of neurons, weighted connections, a bias, and an activation function. As shown in equation 1.1, a neuron takes the value of the sum of the inputs X1, X2, X3, X4… Xn. from the connected neurons multiplies it by the weighted connections W1, W2, W3…Wn. and adds it to a bias value. It then passes this value through an activation function (section 1.4.1) to obtain the final value that is passed to the subsequent neurons [20].

$$input = \sum_{i=1}^{n}(x_i w_i) + b \qquad (1.1)$$

Where:

x_i Is an input value:

i = 1, 2, 3 …, n;

w_i Is the connector weight, And b is the bias value, as the bias has its own weight = w_0 [20].

The inclusion of a bias (b) value in a neural network is crucial since it serves as a threshold for a neuron's output value and boosts the model's ability to match the input. Each neuron has a bias that controls whether or not it is activated. During network training, the bias value is learned along with the weights' values. [20]

Each type of neural network has a specific application, multi-layer perceptron, which is used for basic tasks; recurrent neural networks, which are used for text recognition;

and convolutional neural networks, which are used to classify and distinguish objects in images and videos, are among the most well-known.

For different reasons, the majority of those applications use ML and DL [21].

1.4.1 Activation Function

An activation function specifies how a certain neuron will process inputs to produce an output.

Each neuron in a neural network has an activation function that controls how inputs will be processed, as seen in Figure 1.6.

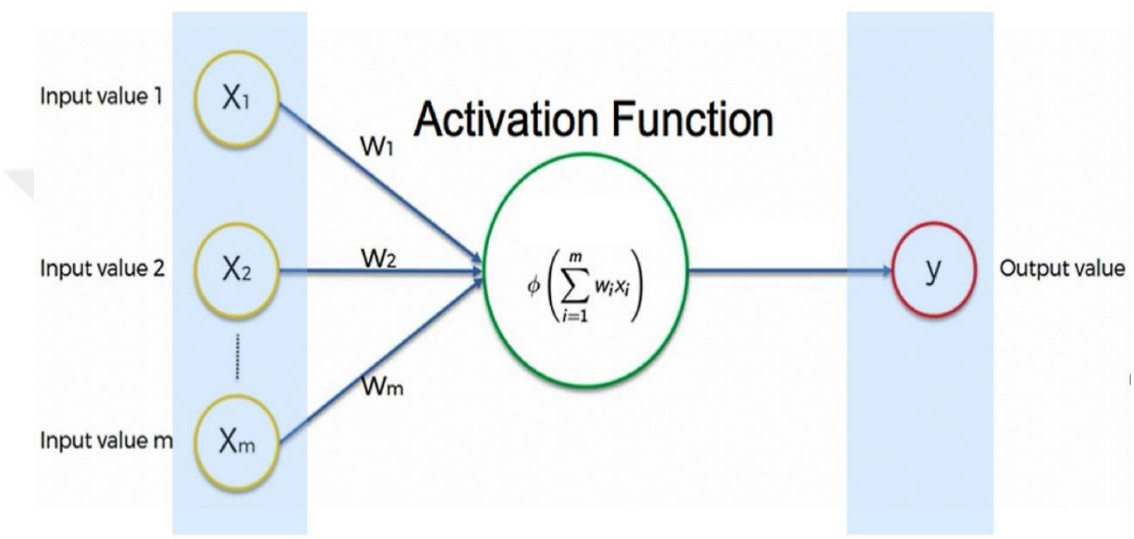

Figure 1.5. A neuron with an activation function[20].

The output receives the results produced by an activation function. The criteria for how the input values should be interpreted to produce an output are set by the activation function. Various activation functions will result in various outputs for the same input data. It's crucial to know how to choose the appropriate activation function when utilizing neural networks to address a particular problem[20].

1.4.1.1 Threshold Function

The threshold function is the most basic form of the activation function. The threshold function returns a binary value, (either 0 or 1). If any of the inputs are greater than 1, it will produce one as the output versus 0 if the input is less than one[20].

As can be seen in Figure 1.7. The output (y) changes to 1 as soon as any signs of life are seen in the weighted sums of the inputs. The threshold activation function is extremely sensitive as a result. Due to a glitch or some noise, it is quite susceptible to being incorrectly triggered by the least significant signal in the input[20].

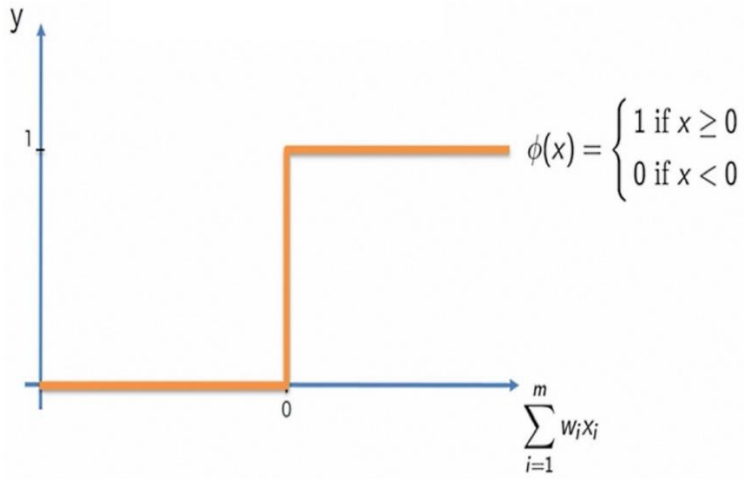

Figure 1.6. Threshold activation function.

1.4.1.2 Sigmoid

We can think of the sigmoid function as the threshold function enhanced. Here, the sensitivity of the activation function can be controlled[20].

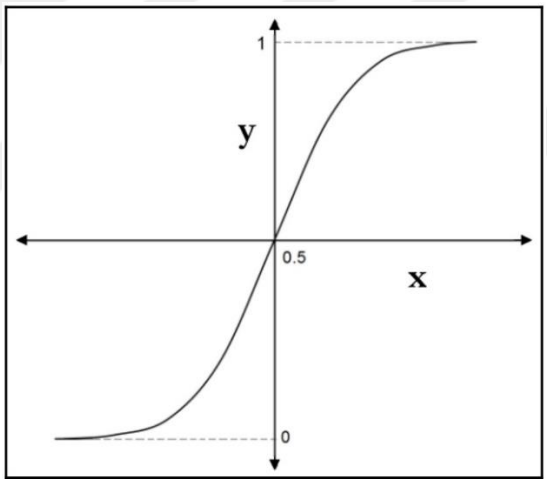

Figure 1.7. Sigmoid activation function.

The following is a definition of the sigmoid function:

$$y = f(x) = \frac{1}{1+e^{-x}} \tag{1.2}$$

1.4.1.3 Rectified Linear Unit (ReLU)

One of the most popular activation functions in neural networks is the ReLU function. The first two activation functions seen in this section have binary outputs, which means they take a collection of input variables and translate them into binary outputs.

While the ReLU activation function transforms a collection of input variables into a single continuous out, ReLU is the most often used activation function in neural

networks and is typically applied in the hidden layers, where we don't want to convert continuous variables into category variables.

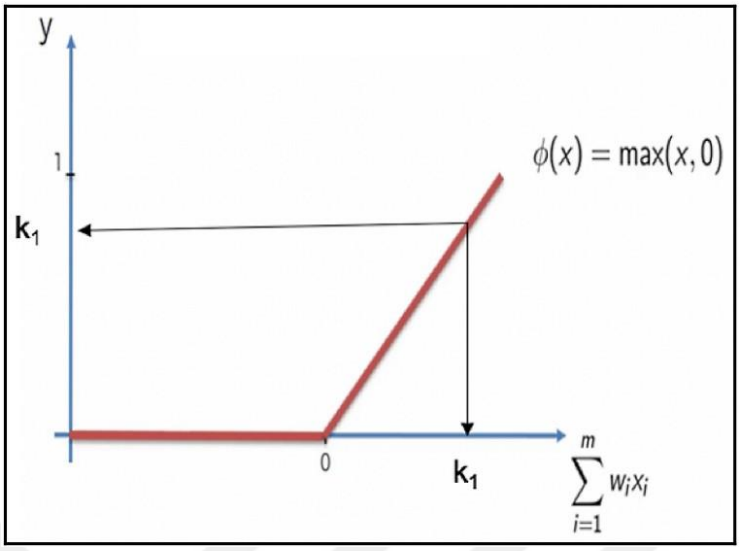

Figure 1.8. ReLU activation function.

When $x \leq 0$, that means $y = 0$, this means that any signal from the input that is zero or less than zero is translated into a zero output.

As soon as x becomes more than zero, it is x. As it is shown in equation (1.3)[20].

$$y = f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{1.3}$$

1.4.1.4 Leaky ReLU

Leaky ReLU is an activation function based on ReLU, but In ReLU, a negative x value yields a zero y value. It indicates that some information is lost during the process, lengthening training cycles, particularly at the beginning of training.

The Leaky ReLU activation function fixes this issue. Rules applied for Leaky ReLU, in Figure 1.10. And Equation (1.4)

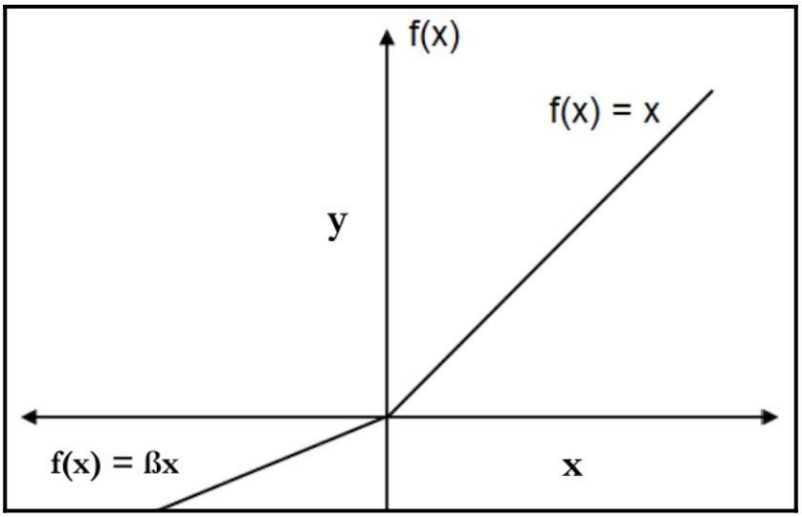

Figure 1.9. Leaky ReLU function.

$$y = f(x) = \begin{cases} ßx, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (1.4)$$

Where ß is a parameter with a value of less than one.

There are three ways of specifying the value for ß:

1. Specifying a default value for ß.
2. Make ß as a parameter in our neural network and let the neural network decide the value (parametric ReLU).
3. Choose a random value for ß (randomized ReLU)[20].

1.4.1.5 Hyperbolic Tangent (Tanh)

The Tanh function is similar to the sigmoid function, while it can also produce a negative signal [20]. Figure1.11. Exemplifies this.

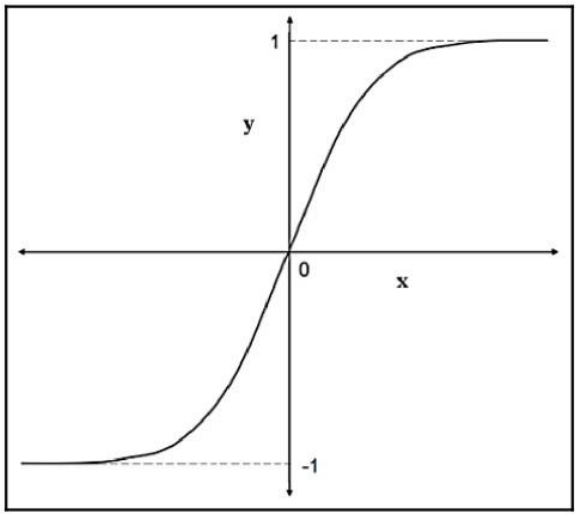

Figure 1.10.Tanh function.

The y function is represented as follows:

$$y = f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \quad (1.5)$$

1.4.1.6 Softmax

For the output of the activation function, more than two levels are occasionally required. An activation function called Softmax gives us more than two levels of output. It functions best in multiclass classification issues. For n input values. Following is how the input values map to the classes:

$$x = \{x^1, x^2, x^3, x^4 \dots x^n\}$$

The Softmax function is Based on the probability theory; Equation (1.6) determines the output probability of the Softmax's e^{th} class [20]

$$prob^{(s)} = \frac{e^{x^s}}{\sum_{i=1}^{n} e^{x^i}} \quad (1.6)$$

1.5 Convolutional Neural Network

Convolutional neural network, often known as CNN or ConvNet, is a DL algorithm that was inspired by biological processes. The structure of connections between neurons in this algorithm is similar to the way that the visual cortex of animals is organized [22].

The CNNs are regarded as a class of DNNs because of their multi-layered construction, which is depicted in Figure 1.12. CNNs use convolution, a linear mathematical operation, instead of matrix multiplications that are used in regular neural networks [23].

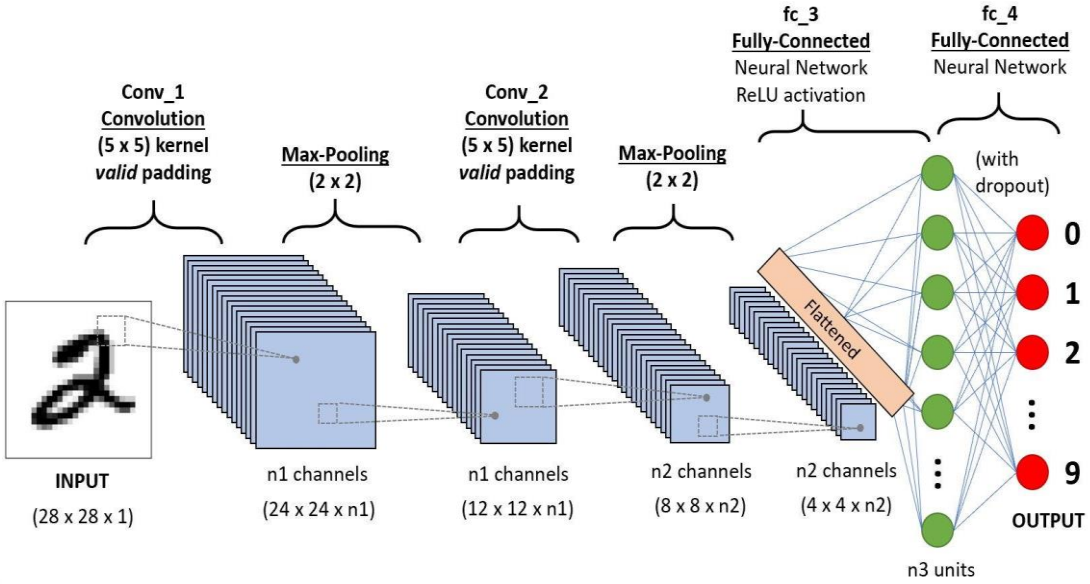

Figure 1.11. The general architecture of CNN [24].

It is regarded as the best object identification and recognition algorithm and, like other DNNs, relies on big data sets for training and for producing more accurate results. The first "Neocognitron" CNNs were introduced in 1980 [22], but their breakthrough came in 2004 when it was demonstrated that neural networks might be accelerated via a graphic processing unit (GPU) rather than a central processing unit (CPU) [15].

With the development of AI and the use of CNNs, computer vision has advanced significantly since it has replaced classical image classification/recognition methods. In contrast to other algorithms, CNN learns the filters and automatically retrieves the features that need to be explicitly programmed [24].

A typical CNN architecture consists of convolutional layers that apply convolution operations (filters) to the input image, creating a feature map. Next, an activation function, such as ReLU, is used to increase non-linearity. Finally, a pooling layer, such as max pooling, is applied, which shrinks the size of neuron clusters and turns them into a single neuron. A fully connected neural network is used to process the features once the pooled images have been flattened into a single vector. The final network will be prepared to cast a vote on the image class that is intended to look for after being trained using forward and backpropagation across multiple epochs[24].

1.6 Face Recognition

The human recognition system generally makes use of a wide variety of stimuli, including those that come from most, if not all, of the senses (such as the visual, aural, olfactory, tactile, etc.). Both the saving and retrieval of face images for the purpose of recognition utilize these stimuli, either individually or together; even if a human can recognize faces without much difficulty, computerized face recognition is still quite difficult to achieve [25].

Over the past several decades, face recognition has been a significant problem in computer vision and pattern recognition.

The handling of differences in expression, position, and illumination when there are just a small number of training samples is one of the challenges in face recognition [26].

Convolutional neural networks (CNNs) have made tremendous advancements in the field of vision, significantly raising the standards for classification challenges. It primarily gains from the end-to-end learning system and the large-scale training data [27]. The most popular CNNs learn features and predict labels, mapping input data to deep features (the result of the last hidden layer, and then to the predicted labels, as shown in Figure 1.13. The classes of the potential testing samples in the generic object, scene, or action recognition are found in the training set, also known as close-set identification. As a result, predicted labels dominate the performance, and Softmax loss is capable of solving the classification issues directly [28].

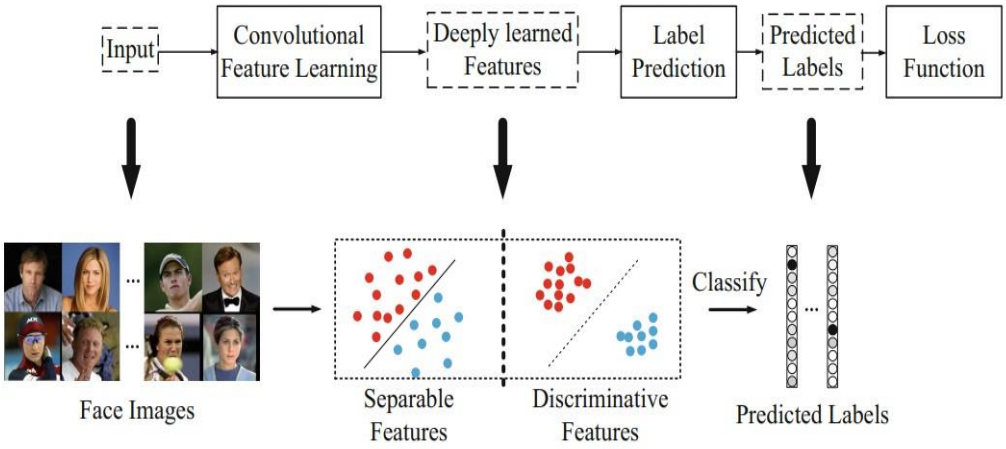

Figure 1.12. Typical convolutional neural network architecture.

Face detection, face alignment, feature extraction, and finally, face recognition are the first four steps in many descriptions of the process of face recognition. [29]

1. Face detection: Determine which faces are present in the image and mark them with a bounding box.
2. Face alignment: Normalize the face so that photometric geometry is in line with the database.
3. Feature extraction: Extract from the facial features that can be applied to the task of recognition.
4. Face recognition: Match the face against one or more known faces in a database that has been prepared.

1.7 Object Detection

Object detection is a branch of computer vision and image processing. The main goal of object detection is detecting instances of semantic objects of a specific class, such as people, buildings, or cars, in digital images and videos [30]

Image classification/localization, target tracking, semantic segmentation, and instance segmentation are the primary functions of computer vision. Deep learning has recently made strides in the study of picture categorization, which has sped up the development of object detection. Existing domain-specific image object detectors typically fall into one of two kinds of object detectors: One-stage detectors like R-CNN [32]. Two-stage detectors like YOLO [33] and SSD [34].

One-stage detectors achieve excellent inference speed, whereas two-stage detectors achieve great localization and object recognition accuracy. The RoI (Region of Interest) pooling layer can be used to separate the two stages of two-stage detectors. For instance, the initial step of Faster R-CNN is known as RPN or a Region Proposal Network, and it suggests potential object bounding boxes. For the subsequent classification and bounding-box regression tasks, features are extracted using the RoIPool (RoI Pooling) procedure from each candidate box in the second step [35].

The basic structure of two-stage detectors is shown in Figure 1.14 (a), which includes a region proposal network that feeds region suggestions into the classifier and regressor. (b) Illustrates the fundamental design of one-stage detectors, which immediately anticipate bounding boxes from the input image in a backbone network [31].

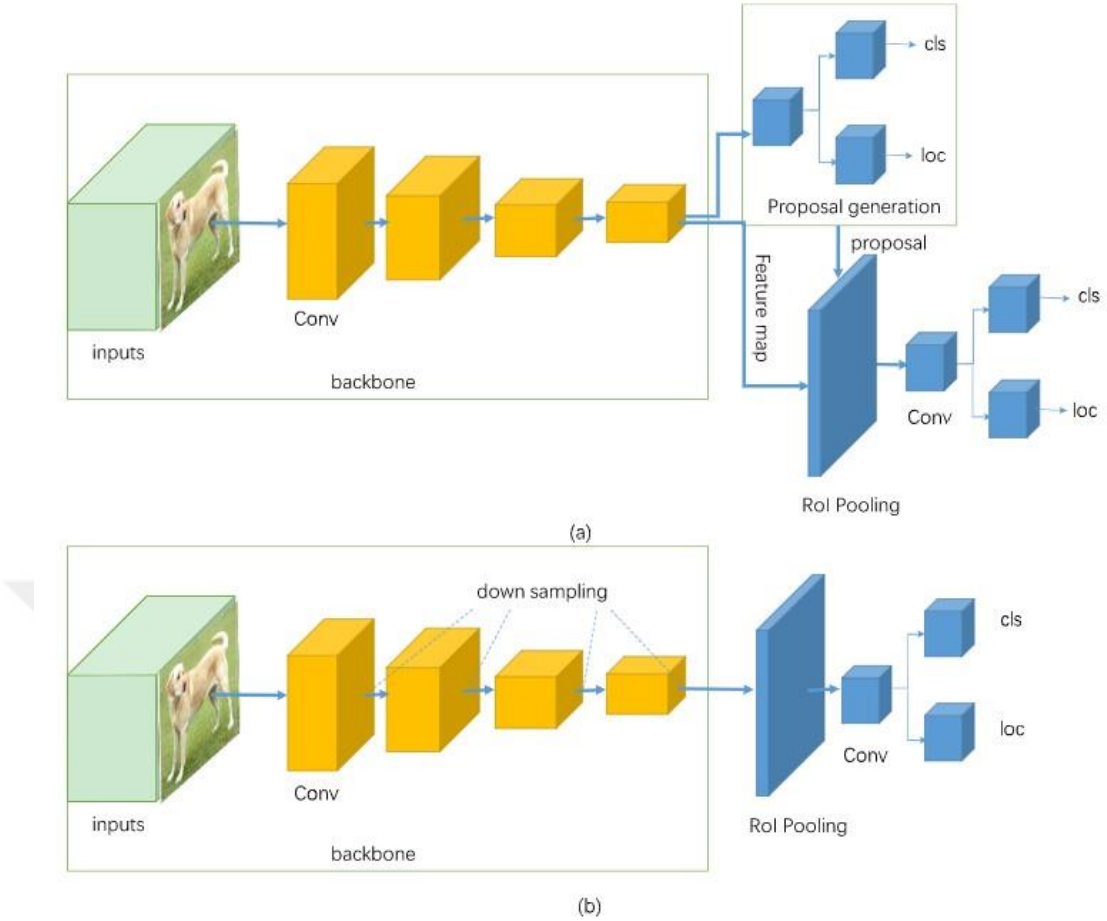

Figure 1.13. The basic structure of both two-stage and one-stage detectors.

Yellow cubes are a series of convolutional layers (called a block) with the same resolution in the backbone net; because of the down-sampling operation after one block, the size of the following cubes gradually becomes small. Thick blue cubes are a series of convolutional layers containing one or more convolutional layers. The flat blue cube demonstrates the RoI pooling layer, which generates feature maps for objects of the same size[31].

1.8 Image Classification versus Object Detection

Identifying the classification of a single object in an image is known as image classification. The process of locating one or more items in an image and tracing a bounding box around their extent is known as object localization. These two tasks are combined in object detection, which locates and categorizes one or more things in an image[36].

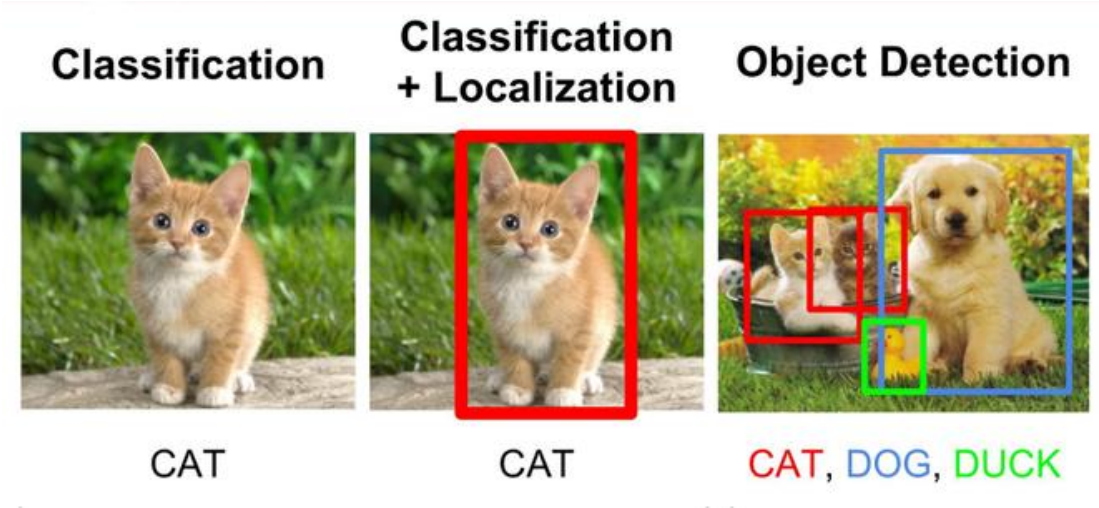

Figure 1.14. Image classifications, object localization, and object detection.

1.9 Problem Statement

People frequently judge others based solely on their outer appearance and then behave with them accordingly. Unconsciously, humans make trait assessments based on appearances, which may significantly affect the outcomes of significant social events like elections and court judgments.

This assumption is made based on facial features ,and an expert is needed to extract them correctly.

Deep learning face recognition technology does not require manual feature extraction, but it also introduced the issue of big data required for training, because its performance is contingent upon the size and the quality of the fed training data.

To overcome the addressed issues, a new dataset is carefully annotated to the needed features specifically for real-time object detection (YOLO) to predict personality traits from face images based on face features(eyes, nose, eyebrows, using YOLOv8 as pa retrained object detection model and higher GPU's to reduce the amount of needed data.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A fast-expanding facial recognition, detection, and feature extraction have made great strides lately as a result of the widespread use of deep learning and computer vision techniques. Due to its potential use in security, surveillance, entertainment, and personal identification systems, it has drawn a lot of attention. Numerous studies have been done as a result to increase accuracy as well as efficacy.

A brief review of several proposed techniques for face detection, feature extracting, and face recognition, presented by previous research, has been presented in this chapter.

S. Z. Li and Juwei Lu [37] (1999). The nearest feature line (NFL), a unique classification technique, is suggested for face recognition.

The feature line (FL) that runs through any two feature points of the same class (or person) generalizes the two points. The capacity of the available database is increased since the derived FL can record more variations of face photos than the original points. The closest distance between each FL and the query feature point is used to classify the data. The NFL error rate varies between 43.7 and 65.4% when using a mixed-face database compared to the traditional Eigenface technique. Additionally, the NFL obtains the lowest mistake rate for the ORL face database that has been recorded to date.

Viola and Jones [38] (2001). presented a technique for rapidly and accurately identifying faces in a picture. This method can be modified to precisely identify facial features. The region of the image being examined for a face feature must be localized to the area that has the best chance of containing the characteristic. False positives are minimized, and the speed of detection is boosted by regionalizing the detection area because the area being inspected is smaller.

D. Nguyen, D. Halupka, P. Aarabi, and A. Sheikholeslami [39] (2006). the new face detection method that is suggested in this work uses a naïve Bayes classifier to identify faces in an image using only edge direction data. This method enables a small FPGA-based implementation that can run at frame rates of over 30 fps in real time. It is also suggested to leverage contrast information from the image to detect the corners, top, and bottom of the mouth using a straightforward lip feature extraction technique. The same FPGA was used for this implementation as well.

M. P. Pamplona Segundo, L. Silva, O. R. P. Bellon, and C. C. Queirolo [40] (2010). Using only depth information as an input, they proposed automatic methods for face segmentation and landmark detection in frontally posed range photos. Edge detection, region grouping, and shape analysis are combined in the segmentation approach to recover the full face region. The method uses surface curvature classification and relief curves of facial photos to identify the inner and outer corners of the eyes as well as the tip of the nose.
In 99.3%, 99.7%, and 96.1% of the photos from the FRGC v1.0, FRGC v2.0, and BU-3DFE databases, respectively, the whole face region was accurately retrieved.

M. Dantone, J. Gall, G. Fanelli, and L. Van Gool [41] (2012). Presented a real-time face recognition system built on the cutting-edge idea of conditional regression forests. These collections of regression trees calculate where various facial landmarks are about the likelihood of certain global face characteristics. By modeling the appearance and position of face feature points conditional to the head pose, they illustrated the advantages of conditional regression forests. On a sizable, difficult database of faces photographed "in the wild," they attained an accuracy comparable to that of human annotators.

Tresadern, P.A., Ionita, M.C. & Cootes, T.F [42] (2012). A real-time facial feature tracking system on a contemporary mobile device was demonstrated: For high efficiency, Haar-like features offer a computationally cheap linear projection, and fixed-point implementations continue to outperform their floating-point counterparts on embedded processors. Hierarchical models that use customized training data also offer excellent accuracy.
Zhenyao Zhu, Ping Luo, Xiaogang Wang, & Xiaoou Tang [43] (2013). For facial recognition, suggested identity-preserving characteristics. The FIP features can be

utilized to recreate facial images in the canonical perspective in addition to being resistant to changes in position and illumination. A deep model with feature extraction and reconstruction layers is used to learn FIP. They also demonstrate that FIP characteristics outperform state-of-the-art face recognition techniques. They also enhanced traditional face recognition techniques.

Y. Sun, X. Wang, and X. Tang [44] (2014).Provide a new technique for learning effective high-level features that disclose identities for face authentication. Deep ConvNets' feature extraction hierarchy is used to summarize the features, which are then built on top of those features. Highly compact and discriminative features are acquired by describing a large number of various identities with a small number of hidden variables. The traits that were taken from various face regions complement one another and improve performance. Using only weakly aligned faces, they were able to obtain 97.45% face verification accuracy on LFW.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao [45] (2016). Proposed a multi-task cascaded CNN-based framework for joint face detection and alignment. Their methods consistently outperform state-of-the-art techniques on several difficult benchmarks, including the FDDB and WIDER FACE benchmarks for face detection and the AFLW benchmark for face alignment, while achieving real-time performance for 640 x 480 VGA images with a minimum face size of 20 x 20, according to experimental results. The meticulously crafted cascaded CNN architecture, the online hard sample mining technique, and joint face alignment learning are the three primary contributions to performance enhancement.

S. Guo, S. Chen, and Y. Li [46] (2016). A convolutional neural network (CNN) is utilized as a feature extractor, and a support vector machine (SVM) is employed as a classifier in this work to present an efficient face recognition system. Optimization methods are used to train CNN in order to enhance its performance. Pre-training CNN with some auxiliary data enhances the network's generalization ability, which results in a significantly faster extraction of facial features from the target dataset. SVM, which has advantages in classification, receives more precise recognition results when the features of the output layer are used as its input.

Muhammad Asim, Zhu Ming, and M. Y. Javed [47] (2017). Proposed an approach that requires no Preprocessing steps like face detection and refining face regions or enlarging the original input images with specific rescaling ratios. Although CNN cannot learn temporal features, spatiotemporal features are crucial for face anti-spoofing.

To extract spatiotemporal characteristics from video sequences and gather the most telling indicators between authentic access and impostor attacks, we cascade LBP-TOP with CNN. Extensive experiments are performed on two extremely difficult datasets, CASIA and REPLAY-ATTACK, which are publicly accessible and produce results with a high competitive score when compared to state-of-the-art approaches.

Zhang, T., Qin, R.-Z., Dong, Q.-L., Gao, W., Xu, H.-R., & Hu, Z.-Y. [48] (2017). They examine whether measured IQ and self-reported personality traits may be predicted from facial photographs in this paper. To handle the task, an end-to-end convolutional neural network is suggested. To assess the prediction, experiments using classification and regression are both runs. The findings indicate that some personality qualities could be reasonably predicted from face photos but that predicting intellect from the face is challenging. Images. Comparative studies reveal that the performance of features derived from face photos using CNN outperforms traditional handmade characteristics that predict attributes.

Li, J., Zhang, D., Zhang, J., Zhang, J., Li, T., Xia, Y., Yan, Q., & Xun, L. [49] (2017). This study employed Faster R-CNN to recognize facial expressions. The benefits of facial expression recognition include the following: The entire network input was taken from the original image. The classic method of facial expression identification avoids the feature extraction step. The features are automatically retrieved from the training dataset by the network. A precise and effective region proposal was created using the Region Proposal Networks (RPNs). The suggested method finds the face region in each image and quickly detects the expression. The results of the experiments demonstrate that the suggested strategy produced improved recognition results.

X. Ziwei et al [50] (2020). To prompt the user to remove the obstruction before face recognition, the SSD network model is used in this study to detect face occlusion. This significantly lowers the difficulty of face recognition, increases its speed and

efficiency of execution, and offers a novel concept for automatic intelligent face recognition.

The detection model adopted in this research can effectively address the detection and classification difficulties and, according to experimental results, can generally meet the requirements for face occlusion detection.

Kumar, A., Kalia, A., Verma, K., Sharma, A., & Kaushal, M [51] (2021). In order to address the face mask identification issue, this study proposes a brand-new, state-of-the-art dataset that is substantially more feature-rich than any existing datasets in the problem domain. The proposed dataset consists of 52,635 photos with more than 50,000 tightly bounding boxes distributed among classes with masks, without masks, wrongly applied masks, and mask area, which helps produce findings that are more precise and dependable than those produced by state-of-the-art methods, To perform face mask detection, the dataset has been tested and validated on the original and small variations of YOLO. With an mAP of 71.69%, the results demonstrate that the original YOLO v4 is the most accurate object detection system.

Nguyen Quoc, H., & Truong Hoang, V. [52] (2021). In this research, a novel YOLOv3 and RetinaFace-based ear detection system is proposed. The outcomes of the experiments have demonstrated how effectively the method operates. Both inference speed and accuracy have improved over the earlier ear detectors.

Chen, W., Huang, H., Peng, S., Zhou, C., & Zhang, C. [53] (2021). To enhance face detection performance, they suggested the YOLO-face face detector, which is based on YOLOv3. The current method uses a more accurate regression loss function and anchor boxes that are more suited for face detection. The upgraded detector maintained its quick detection speed while greatly increasing accuracy.

2.2 Summary

The deference procedures and methods utilized to construct a face recognition-related challenge are shown in Table 2.1.

Table 2.1. Summary of Literature Review

Objective	**procedure**	**References**
Face recognition	Nearest feature line (NFL)	S. Z. Li and Juwei Lu(1999)
Face features detection	Boosted cascade	Viola, P., & Jones, M(2001)
Real-time face detection and lip feature extraction	naïve Bayes classifier	D. Nguyen,et al (2006)
Facial Landmark detection	edge detection, region clustering, and shape analysis	M. P. Pamplona et al (2010)
Face recognition	conditional regression forests	M. Dantone, J. et al (2012)
Real-time facial feature tracking	Haar-like feature	Tresadern, P. et al (2012)
Facial recognition (identity-preserving)	Deep learning(CNNs)	Zhenyao Zhu, Ping Luo, eetal.2013).
Deep face features extraction	DeepID(CNNs)	Y. Sun, X. Wang and X. Tang(2014)
joint face detection and alignment	Multi-task cascaded CNNs	Zhang, K. et al (2016)
Face recognition	CNN and SVM	S. Guo, S. Chen, and Y. Li(2016)
spatiotemporal feature extraction for face anti-spoofing	CNN	Muhammad Asim. et al (2017)
Physiognomy: Personality traits prediction by learning	CNN	Zhang, T. Et al. (2017)
Facial Expression Recognition	Faster R-CNN	Li, J., Zhang et al (2017)
Face Occlusion Detection	SSD	X. Ziwei et al. (2020)
Face mask detection	YOLO	Kumar, A. et al. (2021)
Real-time ear detection	YOLO	Nguyen Quoc et al (2021).
Face detector	YOLO	Chen, W. et al (2021).

2.3 Research Objective

The main objective of this work is to train a real-time (YOLOv8) face feature recognition that can be used to predict personality traits based on physiognomy for the scope of this research can be summarized in a few lines:

1. Train a real-time object detection model for face-features detection and recognition.
2. Construct a new purpose-built dataset.
3. Build a simple GUI to ease the prediction process.
4. Get the best performance of YOLOv8 in small-size datasets.

CHAPTER 3

DATASET CREATION

3.1 Introduction

For all machine learning tasks, dataset creation is both a time-consuming and challenging operation; specifically, object detection datasets need to be carefully annotated in order to build a good object detection model [54].

Dataset creation goes through multiple stages, including image collecting, image annotation, and exporting the dataset to work on a specific model.

The dataset was first created with 1000 images and a 5k bounding box (version-1), and then a second version of the dataset (version-2) was created, the same as version-1 but with data augmentation to investigate how data augmentation can affect the performance of the YOLOv8 model, a blur of 25px is added to the images,

Another dataset was created (physiognomy2) with less No. of classes to handle the issue of inefficient data amount. In this chapter, a step-by-step dataset creation is presented.

3.2 Image Collecting

Face images for both genders were neatly chosen as the face features are clear and do not have face occlusion (sunglasses, face mask, etc.) from the Flickr-Faces-HQ [55] (which is an image dataset that contains high-quality images of human faces, Under a Creative Commons BY-NC-SA 4.0 license,) The collected images are (1024x1024) resolution, cropped to face only, and stored in Portable Network Graphic format (PNG) Figure 3.1. shows a sample of the images that were collected.

Figure 3.1.sample of collected images.

3.3 Object Labeling (annotation)

The practice of labeling an image or series of images is known as image annotation. Labeling the training dataset is an initial step in the object detection process. This procedure can take a long time depending on the dataset size, the number of classes, and the number of annotations. Computer vision models are only as good as the data that is presented to them, and having appropriate and acceptable labels is a crucial component of giving good data.

There are some different data annotation types (Bounding boxes, Polygonal Segmentation, Semantic Segmentation, Lines and Splines, Key-Point, Landmark, and 3D cuboids,) and it is very important to choose the right annotation type according to its use case[5].

Making precise labels (classes) is followed by making boxes with precise boundaries. Without leaving much room around it or omitting any of the objects of interest, a tight bounding box will precisely fit around the edges of the object to be recognized. This is due to the necessity for computer vision models to be as accurate as possible and learn precisely what constitutes an item without becoming confused with the objects' surrounding. Also, ensuring that all the objects of interest are labeled is a good practice to get high model accuracy.

Many image annotation tools are provided across the Internet; both 'LabelImg' and 'Roboflow annotate' are used to label this dataset. LabelImg is a free, open-source

software tool; Tzutalin launched the software in 2015; it is built with Python and uses QT for its graphical user interface[56]

Roboflow annotate is a self-serve browser-based annotation tool that provides an easy annotation framework and a lot of exporting formats rather than preprocessing and data augmentation options[57].

Figures 3.2. and 3.3. show the user interface for LabelImg and Roboflow, respectively

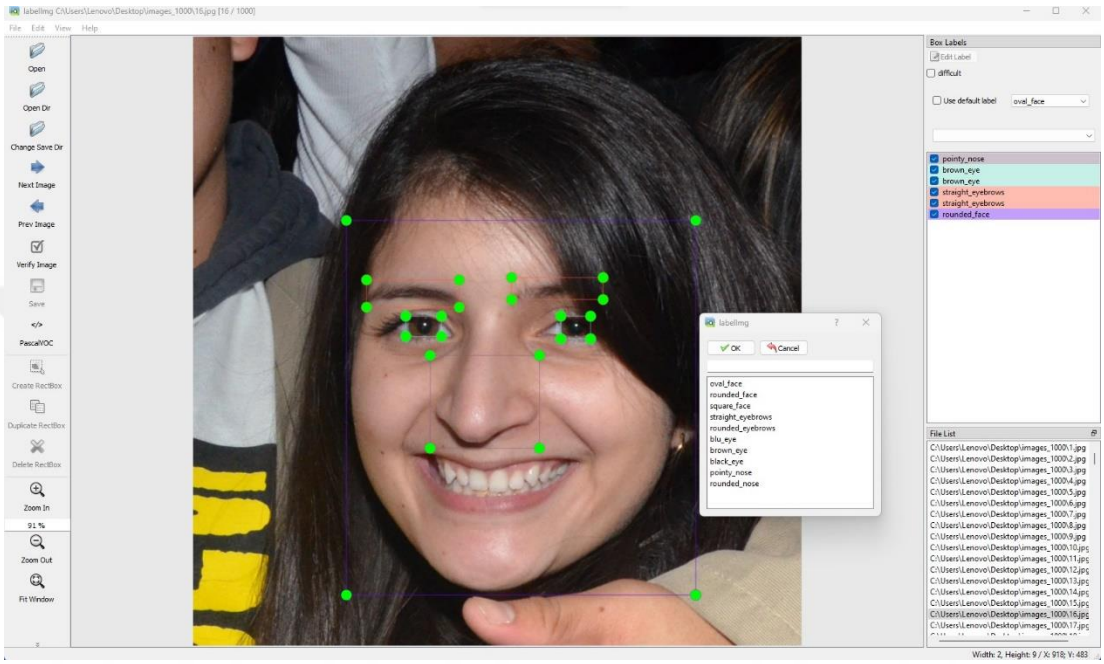

Figure 3.2. LabelImg annotation tool user interface. [56]

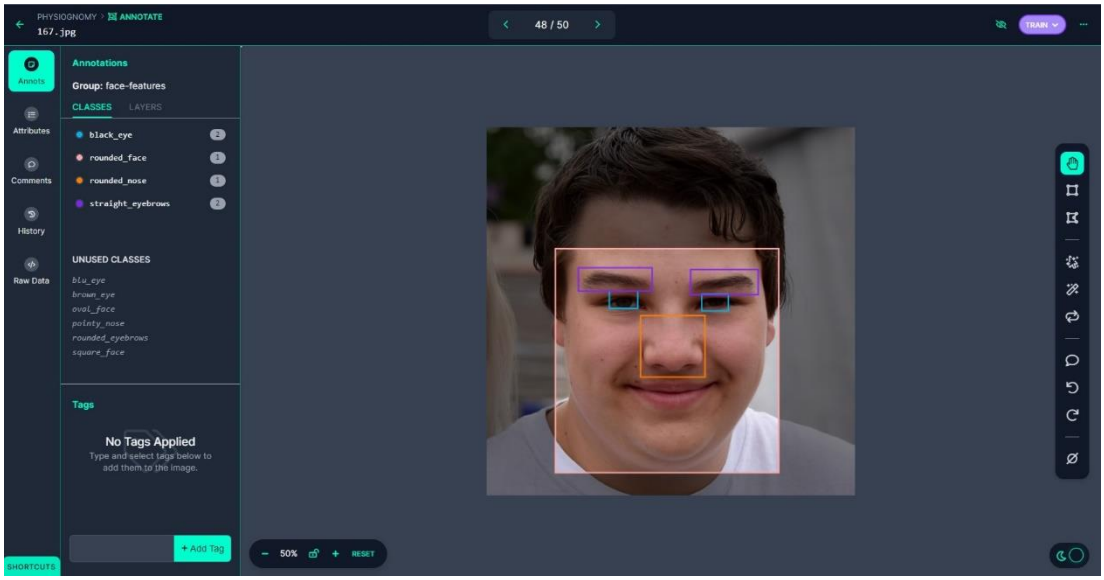

Figure 3.3. Roboflow annotates user interface. [57]

3.4 Exporting Dataset

When it comes to dataset exportat,ion the annotation format need to be taken into consideration for every different object detection model. There are various formats for annotating images, including text files (.txt)and (.csv), TFRecords, image masks, (.json) file for COCO, and Pascal VOC XMLs.[58]

For the YOLO labeling (annotation) format for each image file, a.txt file holds the same name as the image is created, and each (.txt) file provides the object class, object coordinates, height, and width, for each object a new line is created as shown in Figure 3.4 [59].

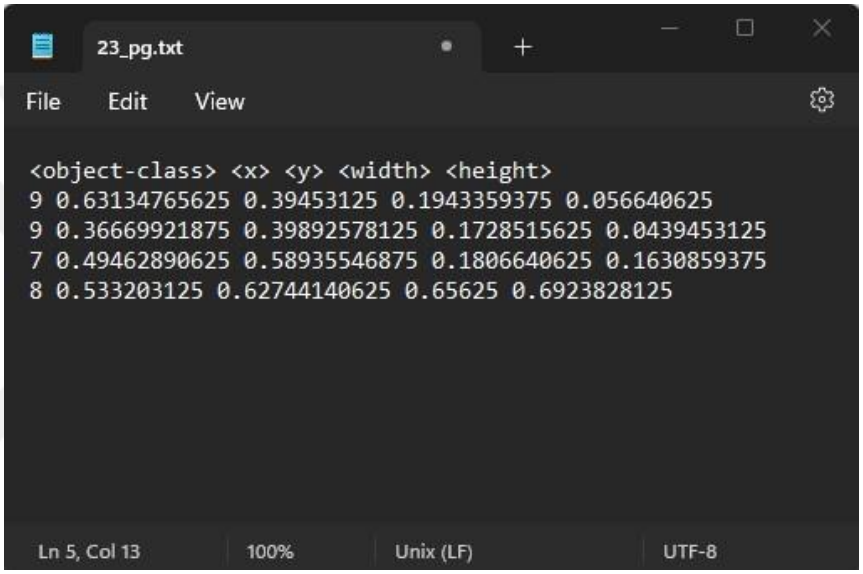

Figure 3.4. YOLO annotation file with four object.

The dataset is then split into three sets (test, train, and validation);every set is in a different folder containing two folders, one for the image and the other for the labels, a data.yaml (Appendix A) is a configuration file containing the train, valid, and test images directory, and the number of classes followed by their names are also found in the main folder. Figure 3.5. show the folder structure of the YOLOv8 dataset.

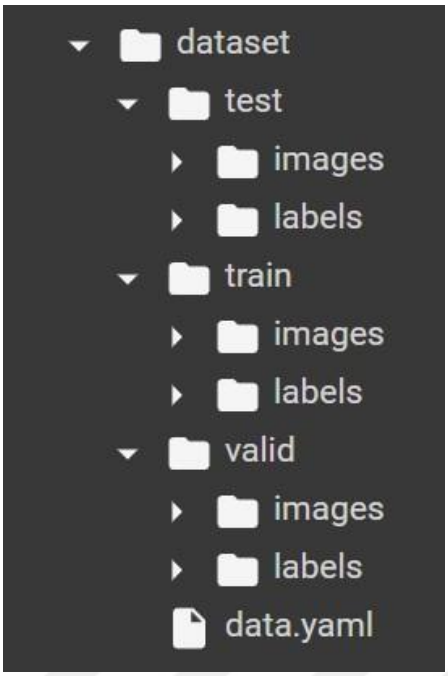

Figure 3.5. Folder structure of YOLO dataset.

3.5 Physiognomy Dataset

Following the fact that there is no dataset available for the scope of this work, a new dataset is created to train a YOLOv8 model.
In detail, class annotation is presented as follows:

1. **Oval face**

   The bounding box is drawn to specify the oval face shape class, and the bounding box drowns with little room to help the model understand the face boundary. According to physiognomy, personality traits linked to an oval face are rational, emotional, highly adaptable, and long-sighted.

2. **Square face**

   The bounding box is drawn to specify the square face shape class, and the bounding box drowns with little room to help the model understand the face boundary.

   According to physiognomy, personality traits linked to a square face are subjective, emulative, frank, honest, serious, and responsible.

3. **Rounded face**

   The bounding box is drawn to specify the rounded face shape class, and the bounding box drowns with little room to help the model understand the face boundary.According to physiognomy, personality traits linked to the rounded face are compassionate, helpful, generous, considerate, honest, and tolerant.

4. **Black eye**

   The bounding box is drawn to specify the black eye class; the bounding box is drawn as tight as possible to the eye's iris. According to physiognomy, personality traits linked to the black eye are trustworthy, responsible, passionate, and lively.

5. **Blue eye**

   The bounding box is drawn to specify the blue eye class; the bounding box is drawn as tight as possible to the eye's iris. According to physiognomy, personality traits linked to the blue eye are youthfulness, peace, kindness, spiritual people, and good observation power.

6. **Brown eye**

   The bounding box is drawn to specify the brown eye class, and the bounding box is drawn as tight as possible to the eye's iris. According to physiognomy, personality traits linked to the brown eye are independent, polite, affectionate, self-confident, determined, persevering, nature lovers, spiritual, and strong-headed.

7. **Rounded eyebrows**

   The bounding box is drawn to specify the rounded eyebrows shape class, and the bounding box is drawn with little room to help the model understand the eyebrow's shape. According to physiognomy, personality traits linked to rounded eyebrows are gentle, wise, kind-hearted, mild-tempered, considerate, and sensitive.

8. **Straight eyebrows**

   The bounding box is drawn to specify the straight eyebrows shape class, and the bounding box is drawn with little room to help the model understand the eyebrow's shape. According to physiognomy, personality traits linked to straight eyebrows are stubborn, masculine, resourceful, decisive, faithful, and upright in their action.

9. **Pointy nose**

   The bounding box is drawn to specify the pointy nose class, and the bounding box is drawn as tight as possible to the nose. According to physiognomy, personality traits linked to the pointy nose are clear thinking, tolerance, trustworthiness, and tenacity.

10. **Rounded nose**

The bounding box is drawn to specify the rounded nose class, and the bounding box is drawn as tight as possible to the nose. According to physiognomy, personality traits linked to the rounded are generous, emotional, kindhearted, and sensitive.

Physiognomy traits are according to [60]; and [61]-[62].

3.6 Dataset Summary

Tables3.1. and 3.2. shows a summary of the physiognomy1 and physiognomy2 dataset respectively.

Table 3.1. physiognomy1 dataset summary.

Parameters	**Value**
Dataset type	Object detection
Annotation type	Bounding box
Images count	1500
classes	10
annotation	8,277
Annotation per image	5.5
Average image size	1.04 MP
resolution	1024×1024
Orientation	square
Format	JPG

Table 3.2. Physiognomy2 dataset summary.

Parameters	Value
Dataset type	Object detection
Annotation type	Bounding box
Images count	1000
classes	6
annotation	10000
Annotation per image	45
Average image size	1.04 MP
resolution	1024×1024
Orientation	square
Format	JPG

This dataset was uploaded to Roboflow named 'physiognomy dataset' [63].

3.7 Comparison with Similar Work

Existing datasets are unsuited for practical facial feature recognition because they typically lack the necessary bounding boxes and annotations. These datasets are more suitable for tasks like face recognition, when it is not necessary to use boundary boxes and annotations, or for face attribute recognition, where specifying the shape and colors of facial features is unnecessary.

This work solves this stoppage by creating an annotated dataset with specific bounding boxes to distinguish facial features based on their shape (e.g., nose, eyebrows) and color (e.g., eyes) to help the process of facial feature detection.

An overview and comprehension of existing datasets and proposed datasets are presented in this section.

Labeled Faces in the Wild (LFW) [64] is an image dataset including face pictures explicitly gathered to study unconstrained face recognition. It contains more than 13K images of faces assembled across the web, where each face is labeled with the person's name in the image; one thousand six hundred eighty of the photographed persons distinctly appear in two or more photos in the data set.

The faces in these images were detected by the Viola-Jones face detector [38]. LFW includes four different groups of images, including the original and three types of aligned images that can be used to test algorithms under other conditions.

CelebFaces Attributes Dataset (CelebA), suggested by [65], CelebA is a huge collection of face characteristics containing over 200,000 celebrity photos, with each one tagged with 40 features. This dataset of photos has a wide range of poses and cluttered backgrounds. The dataset CelebA can be used for computer vision tasks like face attribute recognition, recognition and detection, facial part localization, and face editing and synthesis. CelebA contains a variety of face images with rich annotations, including 10,177 identities, 202,599 photos, five landmark locations, and 40 binary attribute annotations per image. But it doesn't help the suggested issue domain.

Over 3.3 million faces belonging to over 9,000 people are included in MAAD-Face dataset [66]. An extraordinarily bigger number than comparable annotated datasets such as CelebA, which includes 0.2 million faces belonging to 10,000 individuals with just 40 distinct characteristics, and LFW, which has 13.2 thousand faces of 5.7 thousand persons with 74 different features. The amount of feature annotations that MAAD-Face feeds compared to CelebA and LFW is 15 and 137 times more, with 123.9 million feature annotations of 47 extra binary features. Table 1 compares the proposed dataset with the existing dataset in the context of the number of images, classes, and suitability for the proposed problem.

Table 3.3 comperision with related dataset

Dataset		**CelebA**	**LFW**	**MAAD**	**Phisyognomy1**	**Phisyognomy2**
Suitable		NO	NO	NO	YES	YES
Image		0.2M	13.2k	3.3M	1.5K	1K
Classes	**Oval Face**	No	Yes	Yes	Yes	No
	Square Face	No	Yes	Yes	Yes	No
	Rounded Face	No	Yes	Yes	Yes	No
	Black Eye	No	Yes	Yes	Yes	No
	Brown Eye	No	No	No	Yes	Yes
	Blue Eye	No	No	No	Yes	Yes
	Rounded Nose	No	Yes	Yes	Yes	Yes
	Pointy Nose	Yes	No	Yes	Yes	Yes
	Straight Eyebrows	No	No	No	Yes	Yes
	Rounded Eyebrows	No	No	No	Yes	Yes

CHAPTER 4

METHODOLOGY

4.1 YOLOv8

The latest state-of-the-art YOLO (you only look once) model is YOLOv8; it was released in January 2023 by Ultralytics [67], the organization that created YOLOv5. At the time of this writing, there was no official paper about YOLOv8, following the yolov8 GitHub repository and insights into the architectural decisions; compared to YOLOv5, YOLOv8 is anchor-free, which speeds up Non-Maximum Suppression (NMS). YOLOv8 employs mosaic augmentation throughout training, but it is turned off during the final ten epochs because it has been discovered that this augmentation can be detrimental if employed continuously [68].

Five scaled variants were offered by YOLOv8: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large) [67].

4.2 YOLOv8 Model Structure

Object detection models are a class of deep neural networks (or DNN), which means their architecture consists of several hidden layers in addition to the input and output layers. As mentioned before, there is no official paper for YOLOv8; the model architecture is visualized with netro.app [69]. Also, an image provided by a GitHub user, RangeKing, shows a detailed visualization of the network's architecture [70]. As shown in Figure 4.1.

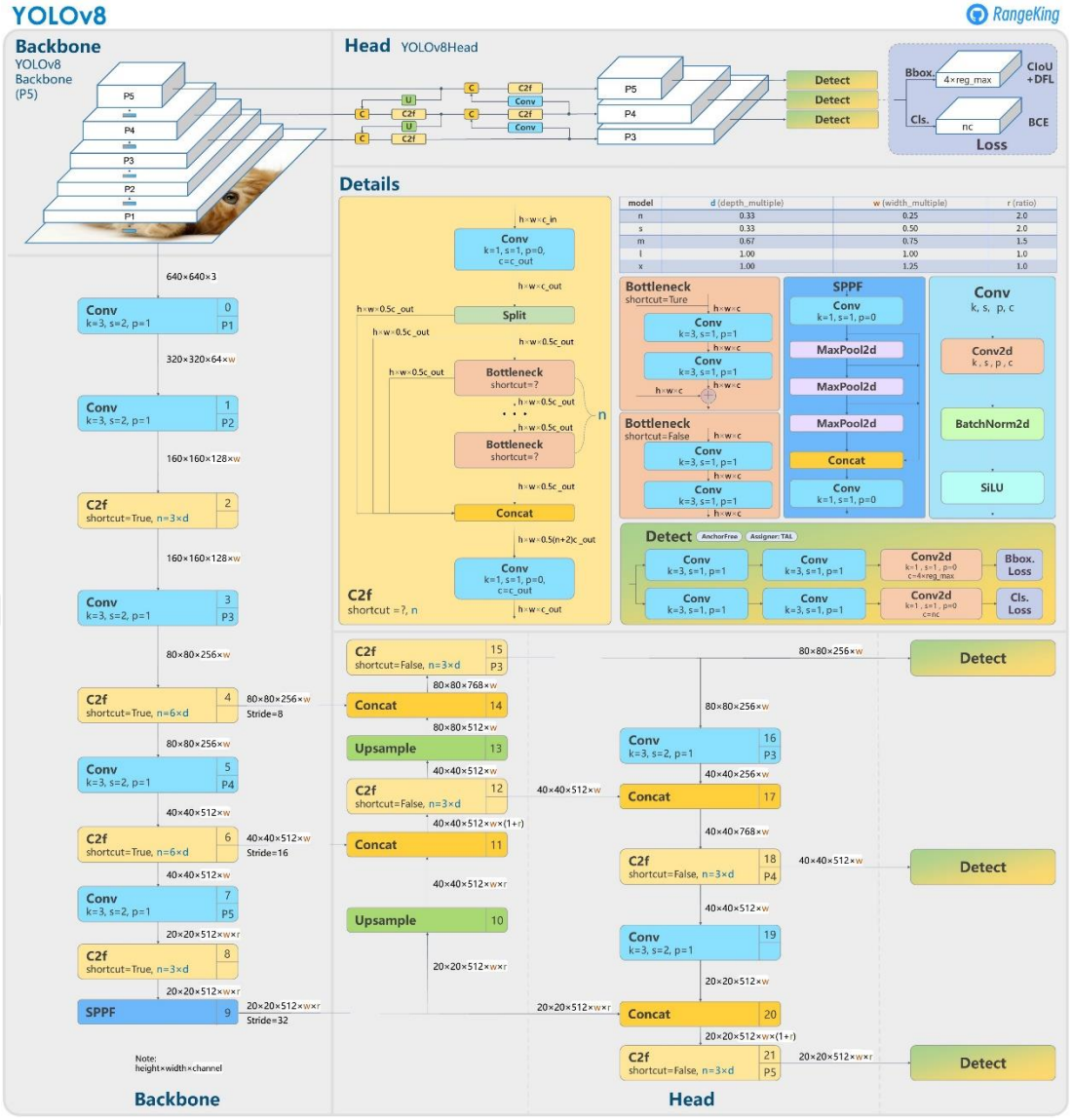

Figure 4.1. YOLOv8 model architecture. [70]

Summarization of the layers and their operation are presented in the following sections.

4.2.1 Conv Layer

As illustrated in Figure 4.1. the Conv layer consists of the Conv2d layer, BatchNorm2d, and SILU activation function.

4.2.1.1 Conv2d layer (2-D convolutional layer): applies sliding convolutional filters to 2-D input; convolution is a mathematical technique that can be compared to combining data from two different sources. Such as in image-related issues, the two inputs were the image itself (three-pixel matrices, one for every channel in the RGB)

and the convolution kernel (small matrix (3 or 5) used for image processing techniques sharpening, blurring, edge detection...etc.) The convolved values (the result of dot production of these two values) form what is known as a ‘feature map’ highlighting a particular feature; the convolutional layer can have multiple feature maps highlighting multiple features [71].

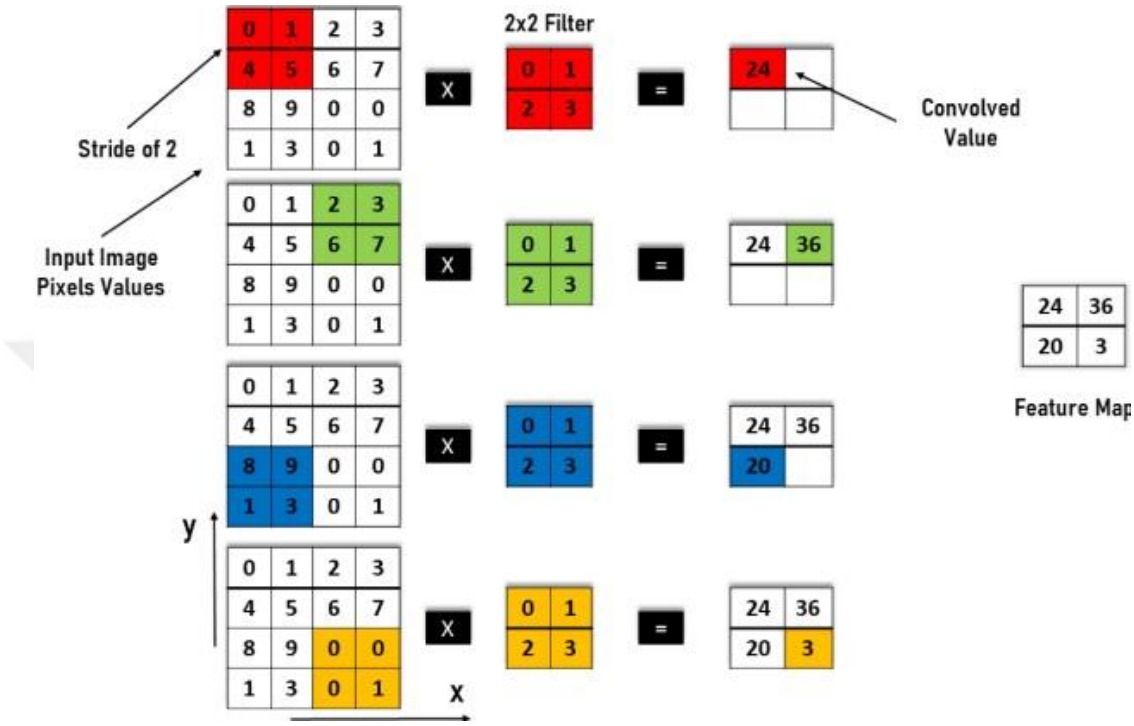

Figure 4.2. How convolution work. [68]

4.2.1. 2BatchNorm2d (Batch Normalization): This is a method for intermediary layers of deep neural networks to normalize activations; batch normalization normalizes the activations of the network between layers in batches so that the batches have a mean of 0 and a variance of 1 to improve the model accuracy and speed up training [72].

4.2.1.3 SiLU (Sigmoid Linear Unit): Sigmoid Linear Unit, also known as the swish function, as the name suggests, and as shown in Figure 4.3. and 4.4. the SiLU activation function is nothing but the multiplication of the sigmoid function with its input [73].

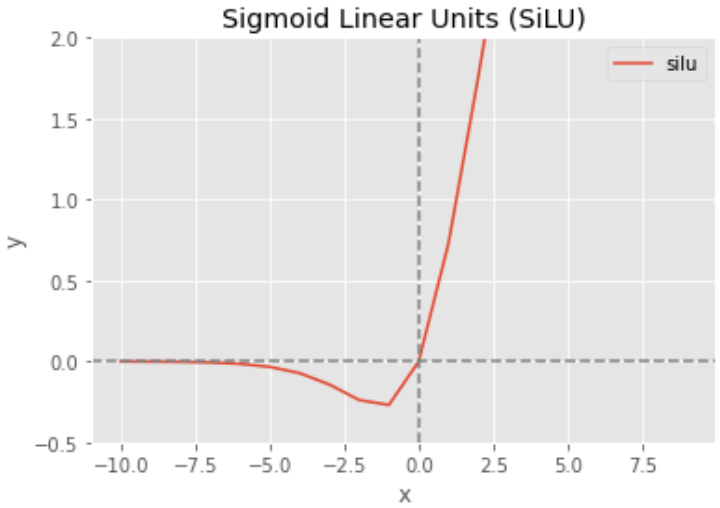

Figure 4.3. SiLU activation function graph. [70]

$$SiLU(x) = x(\frac{1}{1+e^{-x}}) \tag{4.1}$$

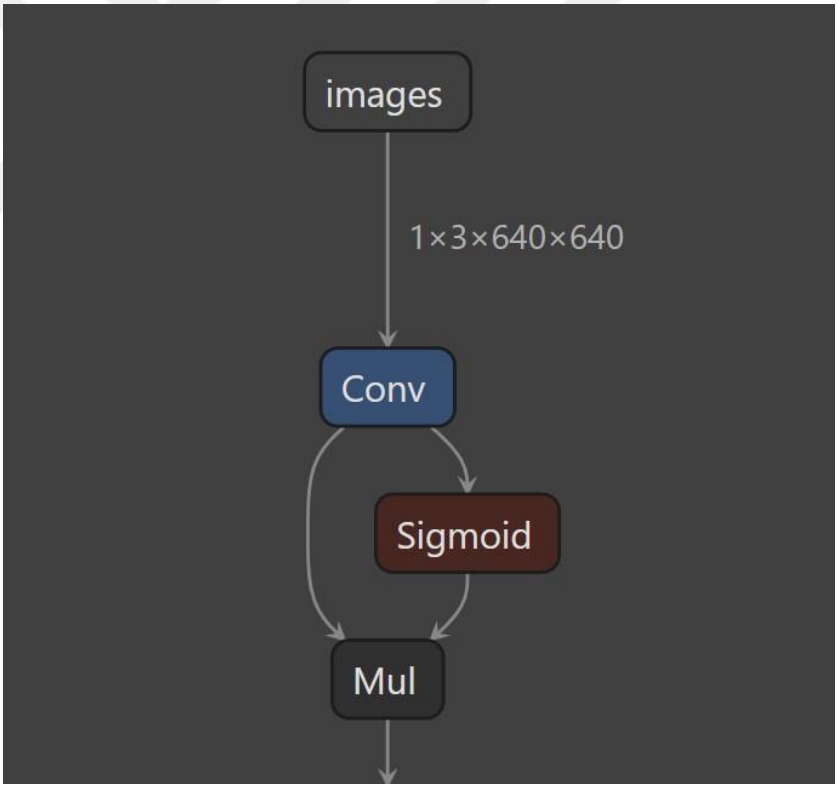

Figure 4.4. Visualization of SilU activation function.

4.2.2 C2f (Coarse-To-Fine)

The Bayesian equation served as the inspiration for the coarse-to-fine layer. The result of coarse classifications is one of the layer's inputs; in both the training phase and the testing phase, it will have a direct impact on the fine classification, which is the layer's

output. [74] As shown in Figure 4.1. and 4.5. the C2f layer consists of a conv layer, split layer, bottleneck, Concat, and conv layer, respectively.

4.2.2.1 Split Layer:

A utility layer called Split divides one input into several outputs.

4.2.2.2 Concat Layer:

A utility layer that combines multiple inputs.

4.2.2.3 Bottleneck Layer**:**

A layer that has fewer nodes than the preceding layers is referred to as a bottleneck layer; as shown in Figure 4.1. the Bottleneck consist of 2 Conv layer, there are two types of Bottleneck (shortcut=True, in this case, the output of the two conv layers is added with their input, shortcut=False, no addition)

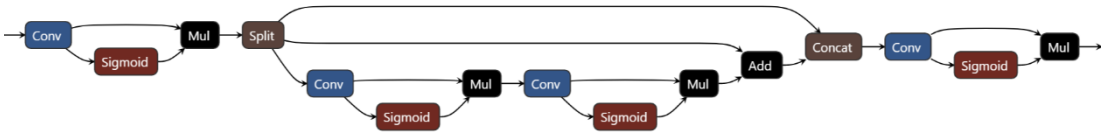

Figure 4.5. Visualization of C2F layer.

4.2.3 SPPF Layer (Fast Spatial Pyramid Pooling)

Fast Spatial Pyramid Pooling is a pooling layer that frees the network's fixed-size restriction. Generally, The Spatial Pyramid Pooling layer pools the features and produces outputs with fixed lengths. [75]. As shown in Figure 4.1. the SPPF layer consists of a conv layer, three maxpool2d layers, Concat layer that combines four inputs; the first one is the output of the conv layer, and the rest are the outputs from each max pool layer, as shown in figure 4.5. and then a conv layer.

4.2.3.1Max pooling layer: A feature map is down-sampled (pooled) by using the Max Pooling operation, which determines the maximum value for patches of the feature map. It is frequently applied following a convolutional layer. It adds a tiny bit of translation invariance, which means that changing the image's size slightly has no effect on the values of the majority of pooled outputs.

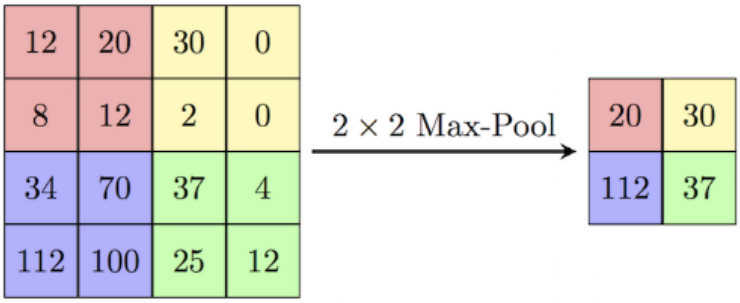

Figure 4.6. Max polling operation.

4.2.4 Detect Layer

In the detection layer, there are two parallel sets of layers consisting of two conv layers and one conv2d layer for bounding boxes and classes as can be noticed in Figure 4.1. and 4.7.

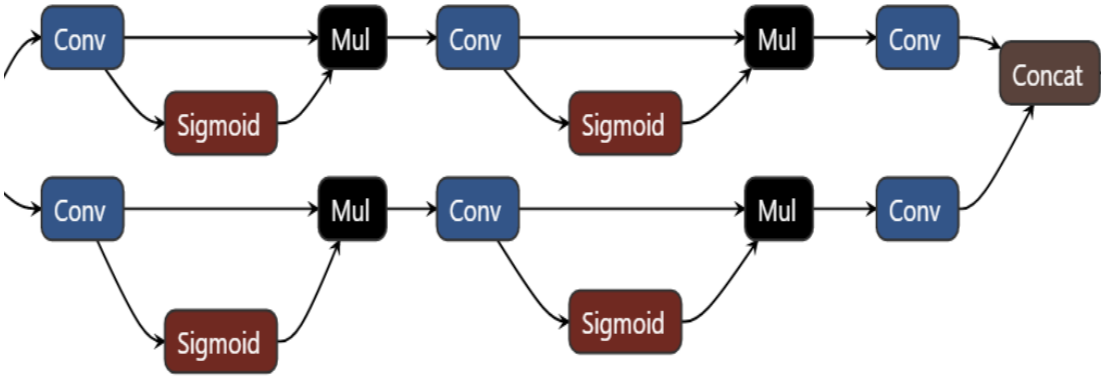

Figure 4.7. Visualization of detect layer.

4.3 Modified YOLOv8

To improve the performance of YOLOv8, a few modifications to the network architecture have been made; the original YOLOv8 model has 365 layers, with 43691520 parameters and 43691504 gradients(Appendix A), and we reduced the depth and width of the network to 0.1 of the actual width and depth also the stride of the second and third convolutional layer of the backbone set to one to increase feature extraction, The resulting modified YOLOv8 is lightweight has fewer layers and requires fewer computation resources, resulting in faster training time, as it has only 211 layers, 2055490 parameters, and 2055474 gradients,(Appendix F) shows the modified model architecture.

4.4 Data Processing

Generally, Data processing is a crucial phase in the YOLO (You Only Look Once) object detection model that involves getting the input data ready for the algorithm's subsequent steps. Usually, the following actions are part of the data processing stage:

1. Resizing of the Input Image: The input Image is scaled to the fixed size that the YOLO model expects (640x640 for YOLOv8). This scaling maintains uniformity across all images and enables the model to process them quickly.
2. The scaled Image may go through further preparation procedures to equalize the pixel values or modify the color channels. Pixel values can be scaled to a certain range, or the image can be converted to a different color space, among other preprocessing methods.
3. Feeding the Image to the Neural Network: The YOLO neural network is then given the preprocessed Image. Multiple convolutional layers are followed by fully connected layers in the network, which work together to evaluate the image and extract features important for object detection.
4. Post-processing and predictions: Following the neural network's processing of the Image, predictions are produced as the output. In order to anticipate bounding boxes, class probabilities, and confidence ratings for each grid cell, YOLO divides the image into a grid. Non-Maximum Suppression (NMS), the most accurate bounding box, can be chosen in order to further enhance these predictions.[76]

4.4.1 Loss Function

The error value in a neural network is calculated using a loss function, which is also referred to as the cost function or objective function. A function that takes the values of the desired output and the predicted probability calculates the error. This measurement aids in defining the network's performance because the network performs better and vice versa, depending on how near the model is to the global minimum of error [77].

In the YOLOv8, the loss is calculated in two parts: the classification loss, calculated using BCE, and the regression loss, calculated using both Focal loss and CIOU, and then the total loss is calculated by adding these losses.

4.5 Evaluation Matrices

The effectiveness of the object detection models such as Fast R-CNN, YOLO, Mask R-CNN, etc., is typically evaluated using mAP or the Mean Average Precision, for set-of detections is the mean over all classes of the interpolated AP for each class, which is given by the area under PR(or precision/recall) curve for the detections, Calculating mAP depends on several sub-matrices such as Confusion Matrix, Intersection over Union (IoU), Recall, and Precision [78].

4.5.1 Confusion Matrix:

The confusion matrix has four attributes.

True Positives (TP): the model predicts a label that matches the ground truth.

True Negatives (TN): The model doesn't predict a label and, there is no label in the ground truth.

False Positives (FP): The model predicts a label and, there is no label in the ground truth.

False Negatives (FN): The model doesn't predict a label and, there is a label in the ground truth.

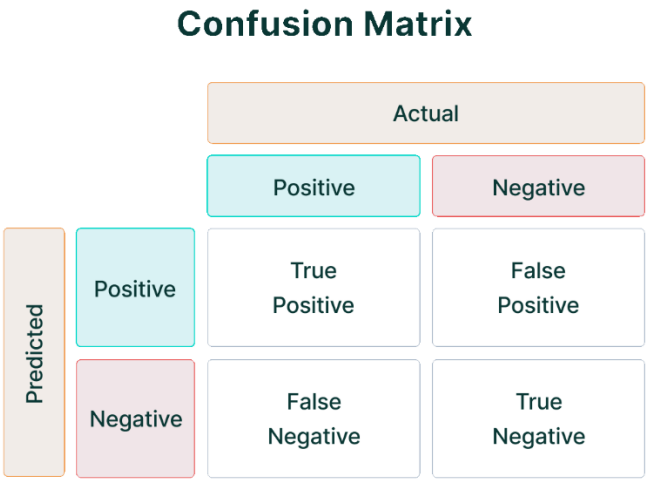

Figure 4.8. Confusion Matrix.

4.5.2 IoU (Intersection over Union):

The IoU calculates the amount of overlap between the ground truth bounding boxes and the predicted bounding boxes.

It's calculated as the intersection area divided by the union area of the two boxes. As shown in equation 4.2[78].

$$IOU = \frac{\text{intersection area}}{\text{union area}} \tag{4.2}$$

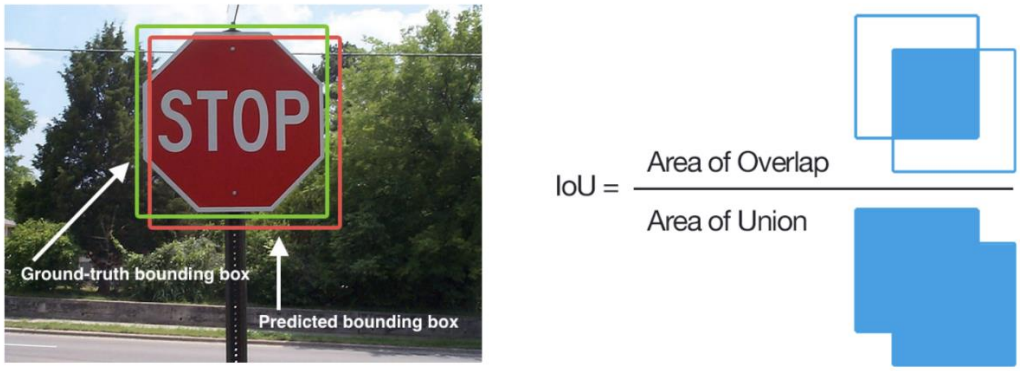

Figure 4.9. How IoU calculated.

4.5.3 Recall:

A recall is a measurement of how well the model can find (TP) out of all predictions, denoted by equation(4.3)[78].

$$recall = \frac{TP}{TP+FN} \tag{4.3}$$

4.5.4 Precision:

Precision is a measurement of how well the model can find (TP) out of positive predictions [78], denoted by the equation:

$$precision = \frac{TP}{TP+FP} \tag{4.4}$$

Obtaining the average precision (mAP) across several classes is done by obtaining the average precision (AP) for each class; for object detection, precision is determined using the IoU threshold, and the precision value varies depending on the IoU threshold[77].

For each class k, the mAP is calculated across different IoU thresholds, and the final metric mAP is calculated by taking an average of all mAP values per class as follows:

$$\text{mAP} = \frac{1}{n}\sum_{k=1}^{k=n} \text{AP}_k \tag{4.5}$$

Where $AP_k$the AP of class k, and n is the number of classes.

4.5.5 F1-Score:

F1-Score: It determines the confidence score threshold with the highest F1-score given by precision and recall. Precision and recall are balanced according to the F1 score. Precision and recall are high when the F1 score is high, and vice versa [78].

$$F1\ Score = \frac{2x(Precision\ x\ Recall)}{Precision + Recall} \tag{4.6}$$

4.6 Model Training

The tools that were employed to train the YOLOv8 model with the new data can be summarized:

Google Colaboratory, or "Colab," is a cloud-based Jupyter Notebook (an open-source web-based development environment Python programming languages, which is a high-level programming language) environment that provides a variety of options when it comes to the hardware component (GPUS and RAM) was used to contain and run the code (Appendix C).

Ultralytics, an object detection API used to get and train the model.

Robowflow is used to download the dataset directly from the labeling workspace.

4.6.1 Hyperparameters

Two different sets of parameters are present in deep learning and machine learning.
The terms learnable parameters and hyperparameters are used to describe them. Learnable parameters, as their name suggests, are obtained and acquired through training.
On the other hand, hyperparameters are factors that influence and aid in managing the learning procedure (or the manner in which the network is trained). The network's weights and biases, which are improved using an optimization technique, can be

summed up with the learnable parameters. Equation (4.9) is used to determine the number of those parameters:

$$N_P = w + b \tag{4.9}$$

Where w is the weight of the node/filter, and b is the bias value.

If the last layer was dense (fully connected), then both the input and output are equal to the number of nodes [79].

in object detection, Hyperparameters can include settings related to the architecture of the model, like the number of the convolutional layers and their size, as well as parameters related to the training process, like the batch size, learning rate, and the number of epochs, hyperparameters might have an impact on how the model behaves, hyperparameters might have an impact on how the model behaves.

The used hyperparameters values in this work are shown in Table 4.1.

Table 4.1. used hyperparameters.

Hyperparameters	Value	description
epochs	200	Number of epochs
patience	25	No. epochs to wait for no observable improvement
batch	16	No. of images per batch
imgsz	640	Input image size
pretrained	True/False	Whether to use a pre-trained model
optimizer	'SGD'	SGD optimizer

4.6.1.1 Batch Size:

The batch size is a hyperparameter that specifies how many samples (images) must be processed before the internal model parameters are updated. The batch is like a for-loop making predictions while iterating over one or more samples (images). The predictions are compared to the anticipated output variables at the end of the batch, and an error is calculated. [80]

4.6.1.2 Epochs:

A hyperparameter called "epochs" determines how many times the learning algorithm will run over the whole training dataset.

Every sample (image) in the training dataset has had a chance to update the internal model parameters once during an epoch. One or more batches make up an epoch. [80]

4.6.1.3 SGD Optimizer:

Stochastic Gradient Descent (SGD) updates a parameter for each training example and label; SGD solved the Gradient Descent issue by updating parameters from just one record. SGD requires forward and backward propagation for every record, which makes it slow to converge. Additionally, the path to the global minima becomes quite noisy[80][81].

4.7 Transfer Learning and Pretrained Model

Transfer learning uses a network that has been trained on a huge dataset, like ImageNet [82], and uses those weights as the starting points for a new classification assignment.[83] Usually, only the weights of convolutional layers, not the entire network with fully-connected layers, are replicated. Since many image datasets have low-level spatial properties that may be learned more effectively with large data, this is particularly useful [84].

Transfer learning and pretraining are conceptually very similar. Pretraining is the process of defining the network architecture and training it using big-size datasets, such as ImageNet [82], and ResNet [85]. Pretraining enables the initialization of weights using big datasets, while still allowing some flexibility in network architecture design [86].

This varies from Transfer Learning in that the weights and network design, such as ImageNet [82] or ResNet [84] are both required in Transfer Learning[81].

Five scaled variants were offered by YOLOv8 as object detection pre-trained models: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large) [64].

Table 4.2. shows a Summary of these pretrained models.

Table 4.2. YOLOv8 pretrained models.

Model	Size(p)	mAP^{50-95}	Speed(ms)	Params(m)	FLOPs(B)
Nano	640	37.3	0.99	3.2	8.7
Small	640	44.9	1.20	11.2	28.6
Medium	640	50.2	1.83	25.9	78.9
Large	640	52.9	2.39	43.7	165.2
Extra large	640	53.9	3.53	68.2	257.8

CHAPTER 5

EXPERIMENTAL WORK

5.1 Experimental Work

As mentioned before, the 8th version of YOLO was presented with five scales; therefore, to have a full understanding of how the physiognomy dataset will perform on each model, Several trials were made to investigate how this dataset will perform on each one using these models' weights as pre-trained model and training from its original yaml (yet another markup language) configuration file, after choosing the best set, a few other trials will be made on different hardware setup (GPUs), and the two versions (augmented and not) of the dataset, to see if faster GPU or data augmentation will affect the model performance. The last trial is on version 3 of the dataset, increasing the training data size.

All the trials will be made on 200 epochs (with patience of 25) and batch size of 16 and 640 input image size; all trials will be shown in detail below.

1. **Trial One**

   The first try is to train the YOLOv8n (nano) model from yaml on version 1 (not augmented) of the physiognomy1 dataset, Table 5.1. shows the used hyperparameter for trial one.

Table 5.1. Trial one hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8n
pretrained	False
GPU	T4

2. **Trial Two**

The 2nd try is to train the YOLOv8n (nano) model using its pre-trained weights on version 1 (not augmented) of the physiognomy1 dataset Table 5.2. shows the used hyperparameter for trial two

Table 5.2. Trial two hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8n
pretrained	True
GPU	T4

3. **Trial Three**

The 3rd try is to train the YOLOv8s (small) model from yaml on version 1 (not augmented) of the physiognomy1 dataset, Table 5.3. shows the used hyperparameter for trial three.

Table 5.3. Trial three hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8s
pretrained	False
GPU	T4

4. **Trial Four**

The 4th try is to train the YOLOv8s (small) model using its pre-trained weights on version 1 (not augmented) of the physiognomy dataset, Table 5.4. shows the used hyperparameter for trial four.

Table 5.4. Trial four hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8s
pretrained	True
GPU	T4

5. **Trial Five**

The 5th try is to train the YOLOv8m (medium) model from yaml on version 1 (not augmented) of the physiognomy dataset, Table 5.5. shows the used hyperparameter for trial five.

Table 5.5. Trial five hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8m
pretrained	False
GPU	T4

6. **Trial Six**

The 6th try is to train the YOLOv8m (medium) model using its pre-trained weights on version 1 (not augmented) of the physiognomy dataset, Table 5.6. shows the used hyperparameter for trial six.

Table 5.6. Trial six hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8m
pretrained	True
GPU	T4

7. **Trial Seven**

The 7th try is to train the YOLOv8l (large) model from yaml on version 1(not augmented) of the physiognomy dataset, Table 5.7. shows the used hyperparameter for trial seven.

Table 5.7. Trial seven hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8l
pretrained	False
GPU	T4

8. **Trial Eight**

The 8th try is to train the YOLOv8l (large) model using its pre-trained weights on version 1 (not augmented) of the physiognomy dataset, Table 5.8. shows the used hyperparameter for trial eight.

Table 5.8. Trial eight hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8l
pretrained	True
GPU	T4

9. Trial Nine

The 9th try is to train the YOLOv8x (x-large) model from yaml on version 1 (not augmented) of the physiognomy dataset, Table 5.9. shows the used hyperparameter for trial nine.

Table 5.9. Trial nine hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8x
pretrained	False
GPU	T4

10. Trial Ten

The 10th try is to train the YOLOv8x (x-large) model using its pre-trained weights on version 1 (not augmented) of the physiognomy dataset, Table 5.10. shows the used hyperparameter for trial ten.

Table 5.10. Trial ten hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8x
pretrained	True
GPU	T4

11. Trial Eleven

The 11th try is to train the YOLOv8n (nano) model using its pre-trained weights on version 2 (augmented) of the physiognomy dataset Table 5.11. shows the used hyperparameter for trial eleven.

Table 5.11. Trial eleven hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-2
Model	YOLOv8n
pretrained	True
GPU	T4

12. Trial Twelve

The 12th try is to train the YOLOv8n (nano) model using its pre-trained weights on version-1 of the physiognomy dataset, using faster GPU (V100); table 5.12 shows the used hyperparameter for trial twelve.

Table 5.12. Trial twelve hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8n
pretrained	True
GPU	V100

13. Trial Thirteen

The 13th try is to train the YOLOv8n (nano) model using its pre-trained weights on version 1 of the physiognomy dataset, using a faster GPU (A100), Table 5.13. shows the used hyperparameter for trial thirteen.

Table 5.13. Trial thirteen hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-1
Model	YOLOv8n
pretrained	True
GPU	A100

14.Trial Fourteen

The 14th try is to train the YOLOv8n (nano) model using its pre-trained weights on (version-3) of the physiognomy dataset, using faster GPU (A100), Table 5.14. shows the used hyperparameter for trial fourteen.

Table 5.14. Trial fourteen hyperparameter.

Hyperparameters	Value
Dataset version	Phsysiognomy1V-3
Model	YOLOv8n
pretrained	True
GPU	A100

15. Trial Fifteen

The 15th try is to train the YOLOv8n (nano) model using its pre-trained weights on the physiognomy2 dataset,

using faster GPU (A100), Table 5.14. shows the used hyperparameter for trial fourteen.

Table 5.15. Trial fourteen hyperparameter.

Hyperparameters	Value
Dataset version	Physiogonomy2
Model	YOLOv8n
pretrained	True
GPU	A100

5.2 Summary

Table 5.15. shows a summary of all trials.

Table 5.16. Summary of 14 trials and their specification.

Trial	Dataset version	Model	pretrained	GPU
1	Phsysiognomy1V-1	YOLOv8n	False	T4
2	Phsysiognomy1V-1	YOLOv8n	True	T4
3	Phsysiognomy1V-1	YOLOv8s	False	T4
4	Phsysiognomy1V-1	YOLOv8s	True	T4
5	Phsysiognomy1V-1	YOLOv8m	False	T4
6	Phsysiognomy1V-1	YOLOv8m	True	T4
7	Phsysiognomy1V-1	YOLOv8l	False	T4
8	Phsysiognomy1V-1	YOLOv8l	True	T4
9	Phsysiognomy1V-1	YOLOv8x	False	T4
10	Phsysiognomy1V-1	YOLOv8x	True	T4
11	Phsysiognomy1V-2	YOLOv8n	True	T4
12	Phsysiognomy1V-1	YOLOv8n	True	V100
13	Phsysiognomy1V-1	YOLOv8n	True	A100
14	Phsysiognomy1V-3	YOLOv8n	True	A100
15	Phsysiognomy2	YOLOv8n	True	A100

5.3 Graphical User Interface (GUI).

A graphical user interface (or GUI) was developed to streamline the image browsing and prediction process because finding an image through code modification is a tedious and time-consuming process. Figure 5.1. shows a screen grab of the GUI program. The GUI window consisted of two buttons: Upload Image, which permits image selection, and Predict, which performs prediction on the imported Image.

PyQt, a GUI module that connects the QT C++ cross-platform framework with the Python language, is used to write the code (Appendix D) for this GUI program.

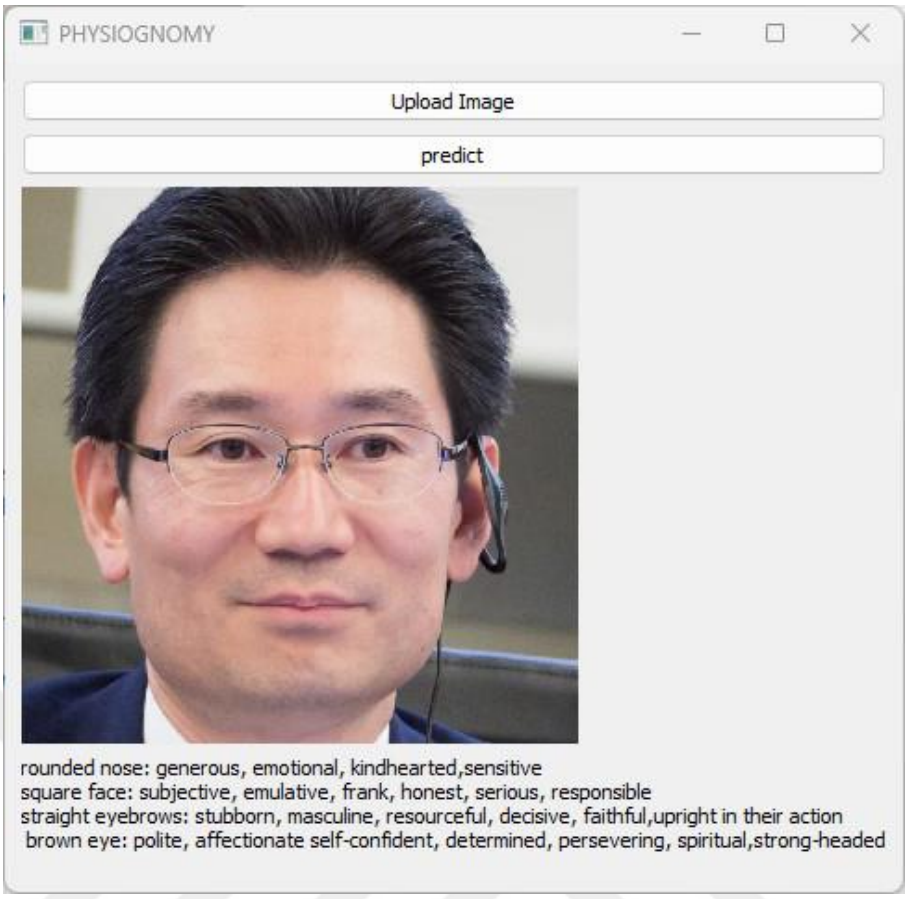

Figure 5.1. Screen grab of the GUI program.

CHAPTER 6

RESULT AND DISCUSSION

6.1 Experimental Results

As discussed earlier, several trials were made to investigate how the dataset performs on YOLOv8; Table 6.1. shows the results of each trial in the context of training time, precession, Recall, the number of epochs needed, and the mAP ^{50}, mAP $^{50-95}$ of all classes(validation).

Table 6.1. Experimantal results.

Trial	epochs	mAP ^{50}(%)	mAP $^{50-95}$(%)	Training time
1	145	78.7	51.6	29
2	75	80	52.4	33
3	116	77.5	50.3	32
4	54	78.1	51.3	14
5	86	78	50.8	41
6	44	79.1	51.6	23
7	65	77.8	50.7	51
8	61	78.2	51.3	45
9	85	76.1	49.9	98
10	46	79.6	51.5	57
11	43	78.5	51.9	24
12	83	78.5	52.5	20
13	73	79.4	52.5	8
14	93	79.7	54.4	14
15	90	90.9	70.3	20

The YOLOv8 model can be considered the best object detection model available with respect to ease of implementation, speed, and accuracy as the model reaches an

mAP $^{50-95}$ of 37.3 for YOLOv8n and 53.9 for YOLOv8x on coco[87] dataset according to [64].

Experimental results show how the proposed model can effectively be used on a custom dataset (physiognomy dataset) to classify the different shapes and colors of facial features. Additionally, the table shows a number of concerns that can be discussed. Firstly, the training time issue as its one of the drawbacks of deep learning in general; training larger model, such as large and extra-large size the model take more time. This issue can be resolved using stronger GPUs by using online cloud services (Google-Colab, AWS,...etc.) that provide development environments with faster GPUs (tesla T4, V100, A100) at reasonable prices, as shown in trial 13. It takes only 8 minutes to train the nano version using A100 GPU compared to Tesla T4 at trial two, which takes up to 33 minutes to train the same model (even slightly better performance with A100 GPU).

The second issue was selecting the proper pre-trained model, and this issue was resolved by tediously testing the five provided model architectures (trials (1-10)) the table shows that the YOLOv8 models gave almost the same result, but the large model was the worst demonstrating that using a bigger model will not necessarily give better performance as the best performance was on the nano version with 52.4 mAP $^{50-95}$ compared to 51.5 mAP $^{50-95}$ using the extra-large.

The third issue was the effect of the dataset size, as through the first 13 trials, the dataset contained 1000 images; increasing the dataset size to 1500 and increasing the training set size led to better performance on trial 14, and another dataset was created (physiognomy2) Reducing the number of classes to 6 (blue_eye, brown_eye, pointy_nose, rounded_nose, straight_eyebrows, rounded_eyebrows) with only 1000 images, the result show a better performance (trial 15) with an mAP ^{50} Of 90% and mAP $^{50-95}$ of 70% as it will be shown in the upcoming figures.

Lastly, there are some challenges during annotating the dataset as it is impossible to get a balanced annotation count as the face has two eyebrows and two eyes, which leads to the number of annotations for eyebrows and eyes is always twice the number of annotations for nose or face shape.

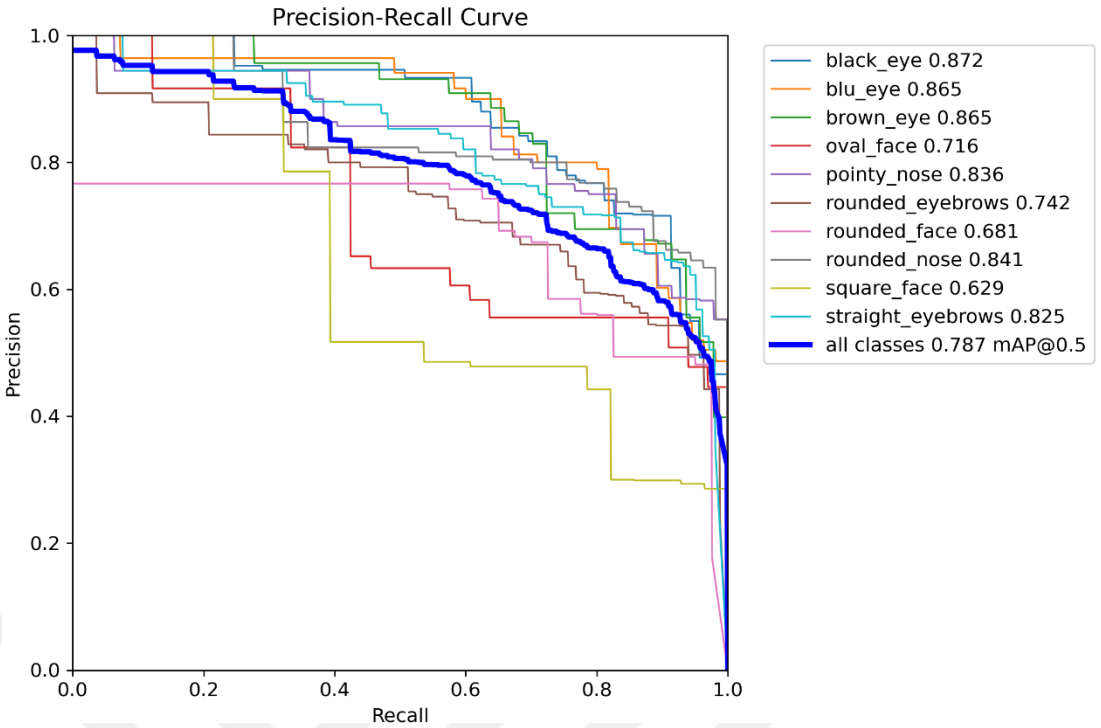

Figure 6.1. P-R-curve of all classes for trial 1.

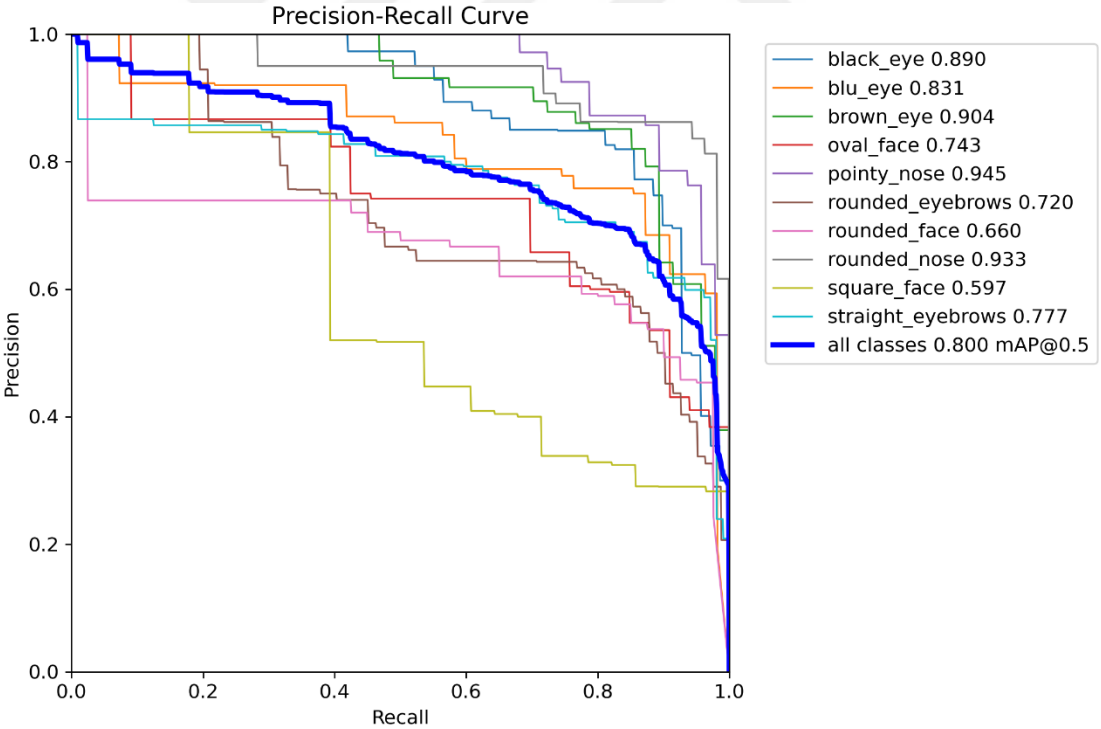

Figure 6.2. P-R-curve of all classes for trial 2.

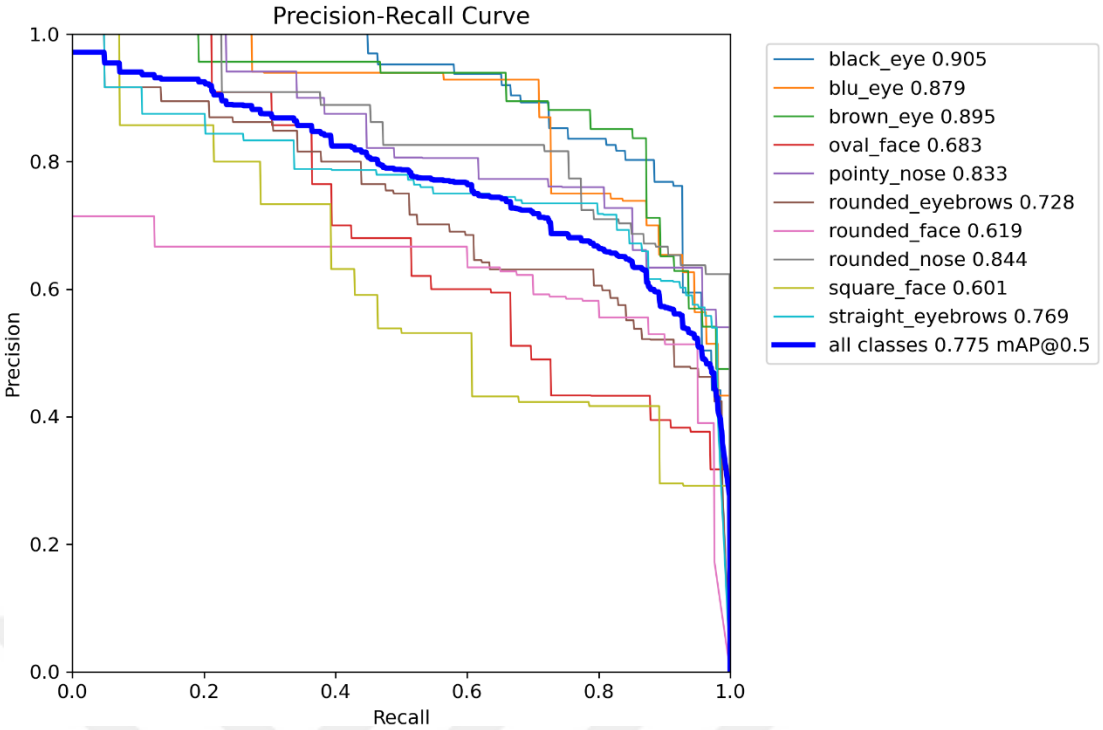

Figure 6.3. P-R-curve of all classes for trial 3.

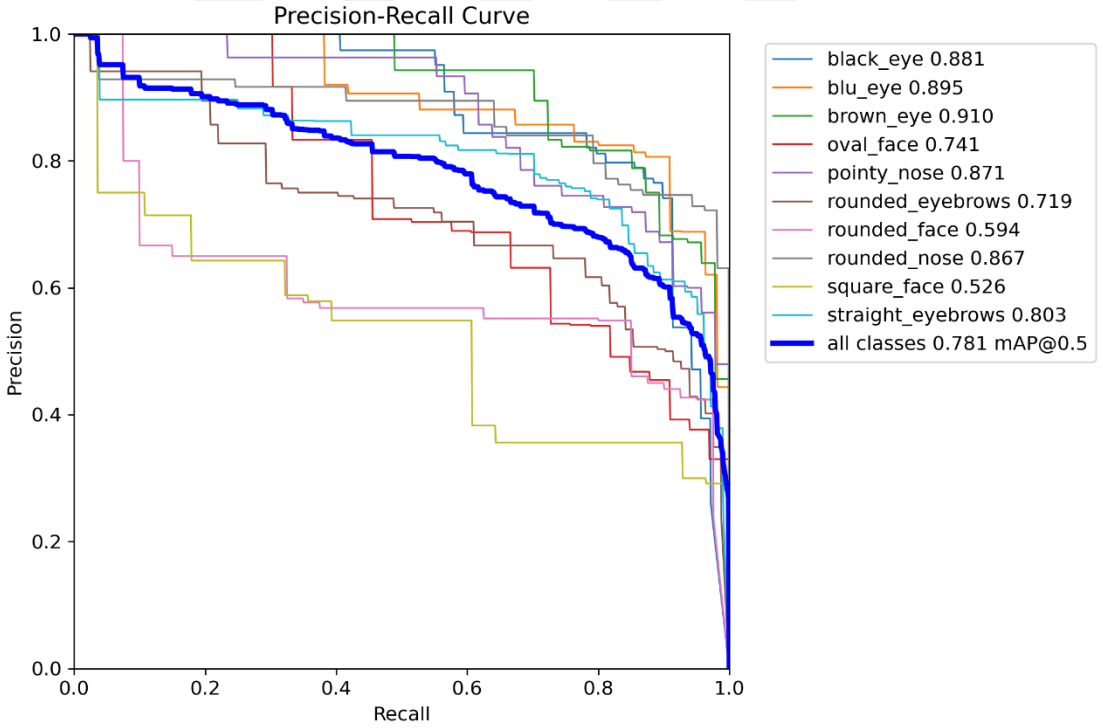

Figure 6.4. P-R-curve of all classes for trial 4.

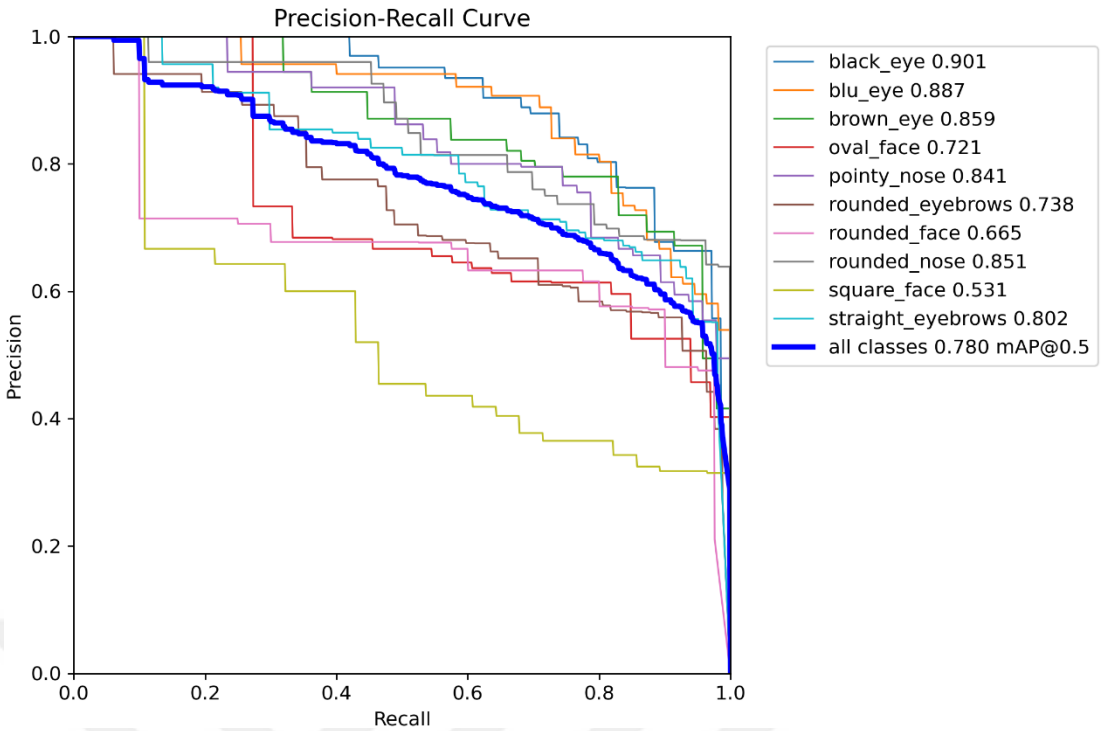

Figure 6.5. P-R-curve of all classes for trial 5.

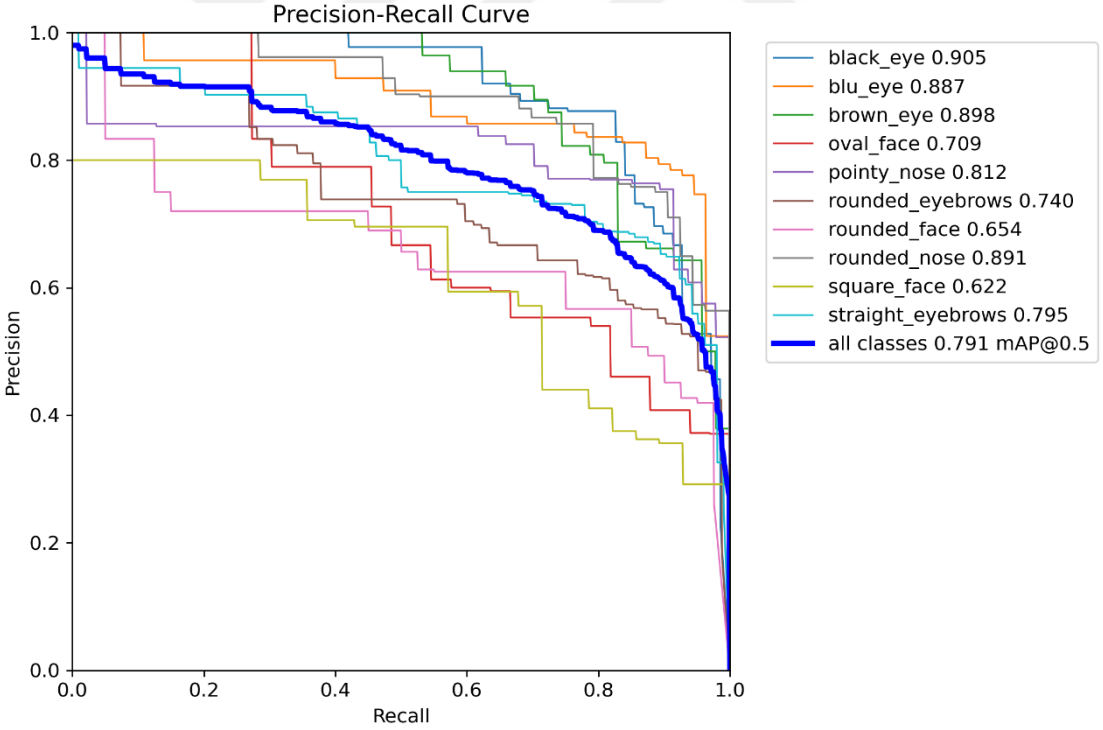

Figure 6.6. P-R-curve of all classes for trial 6.

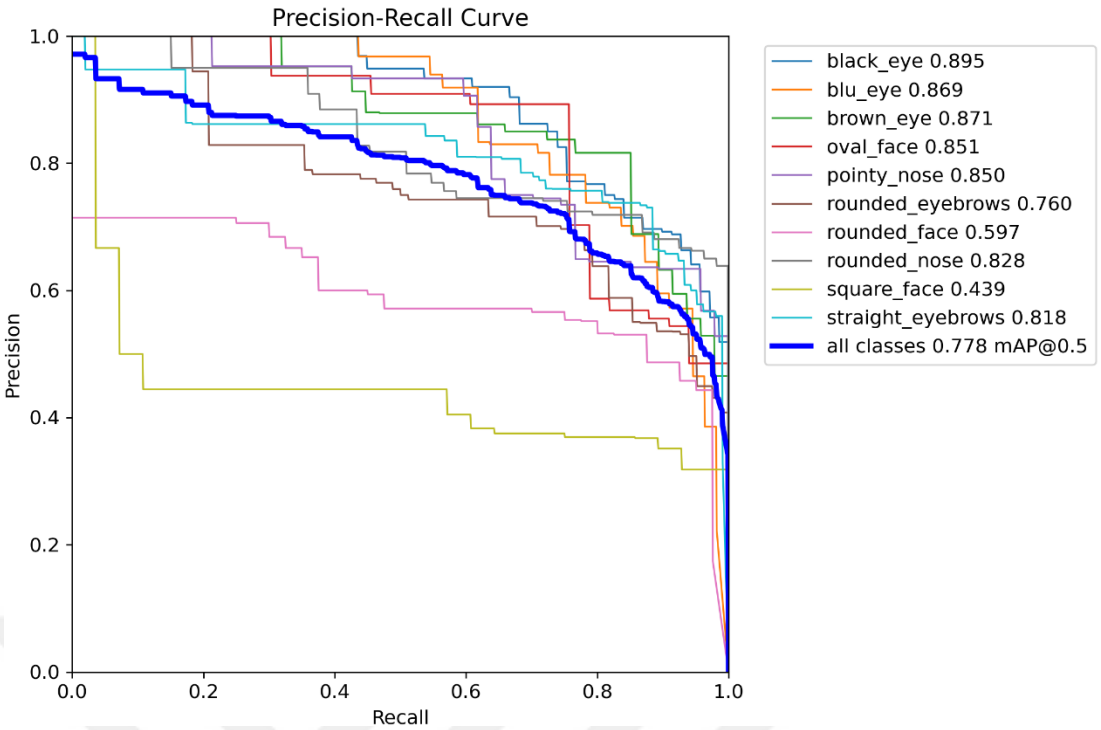

Figure 6.7. P-R-curve of all classes for trial 7.

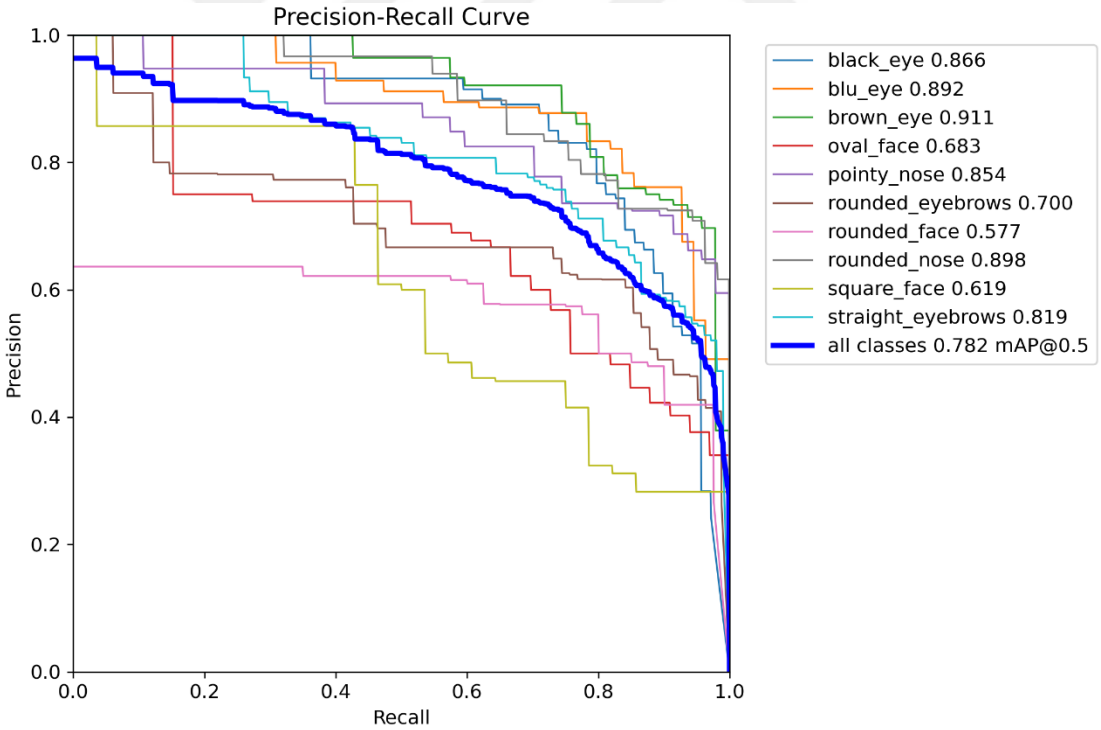

Figure 6.8. P-R-curve of all classes for trial 8.

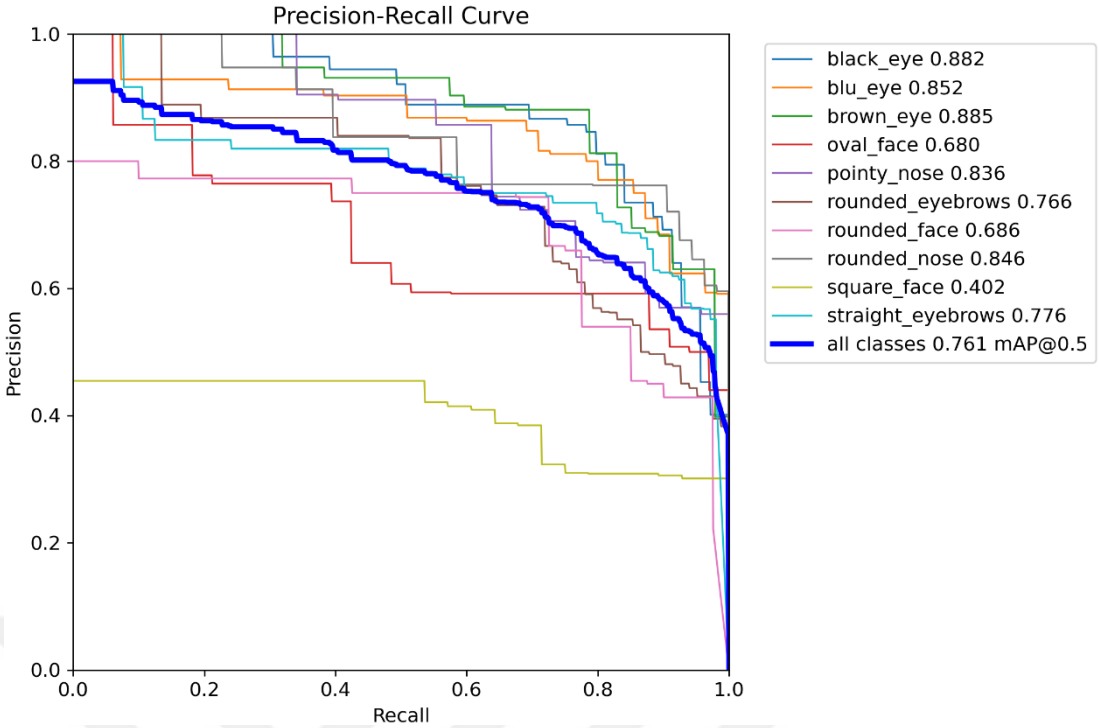

Figure 6.9. P-R-curve of all classes for trial 9.

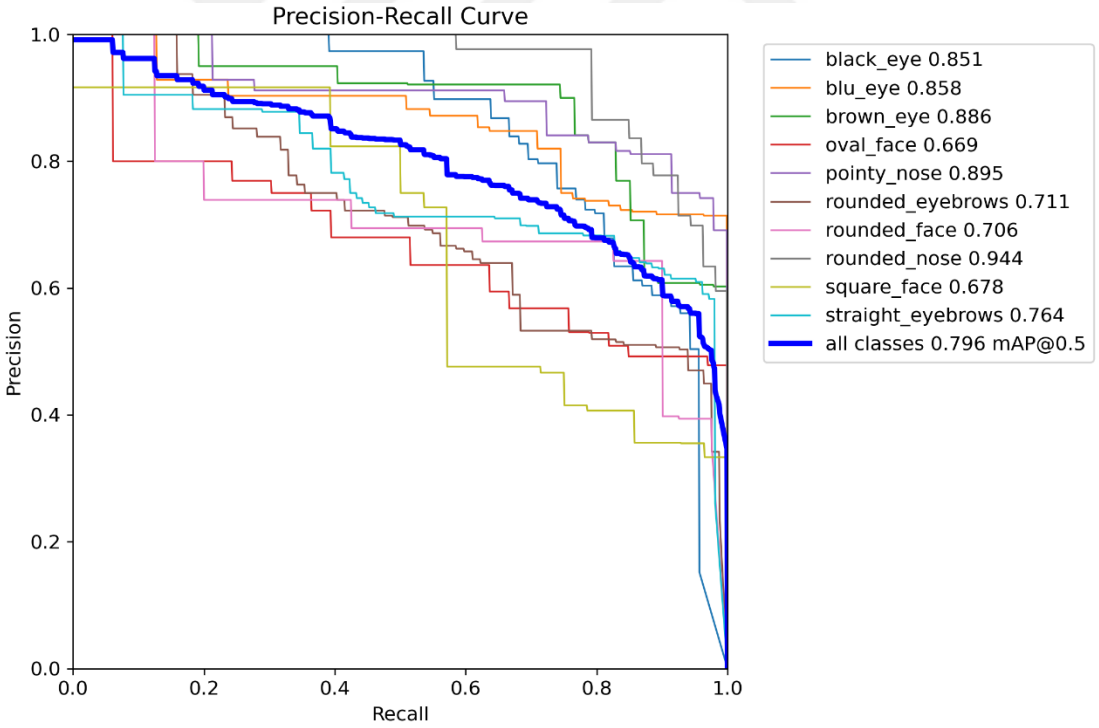

Figure 6.10. P-R-curve of all classes for trial 10.

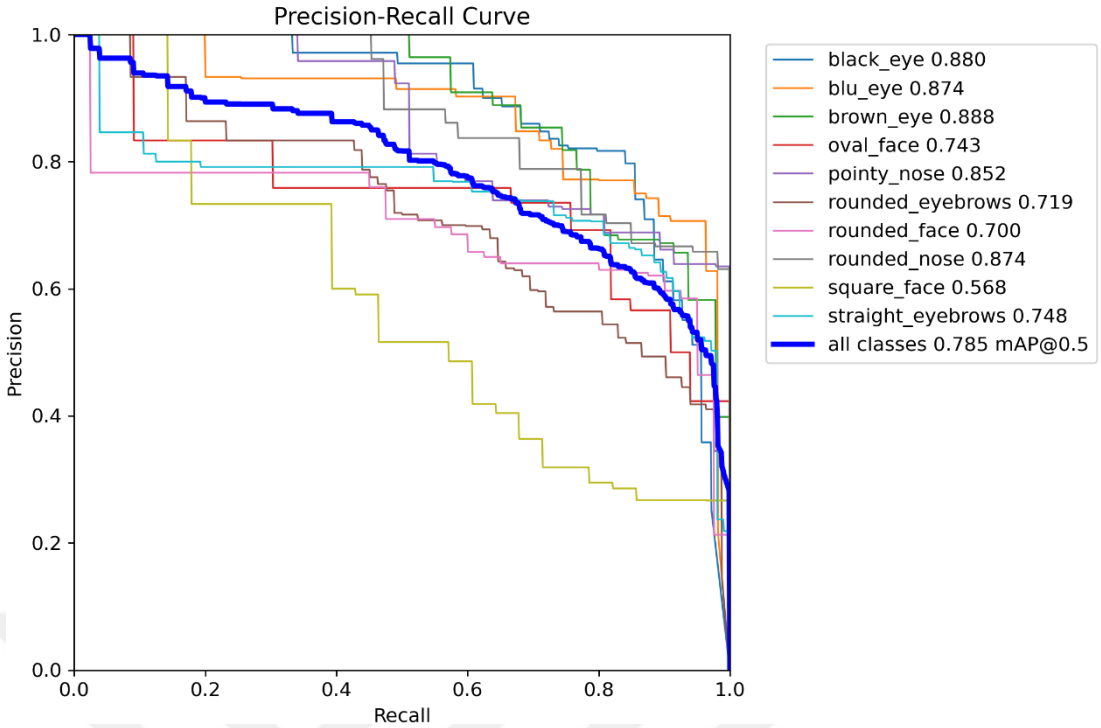

Figure 6.11. P-R-curve of all classes for trial 11.

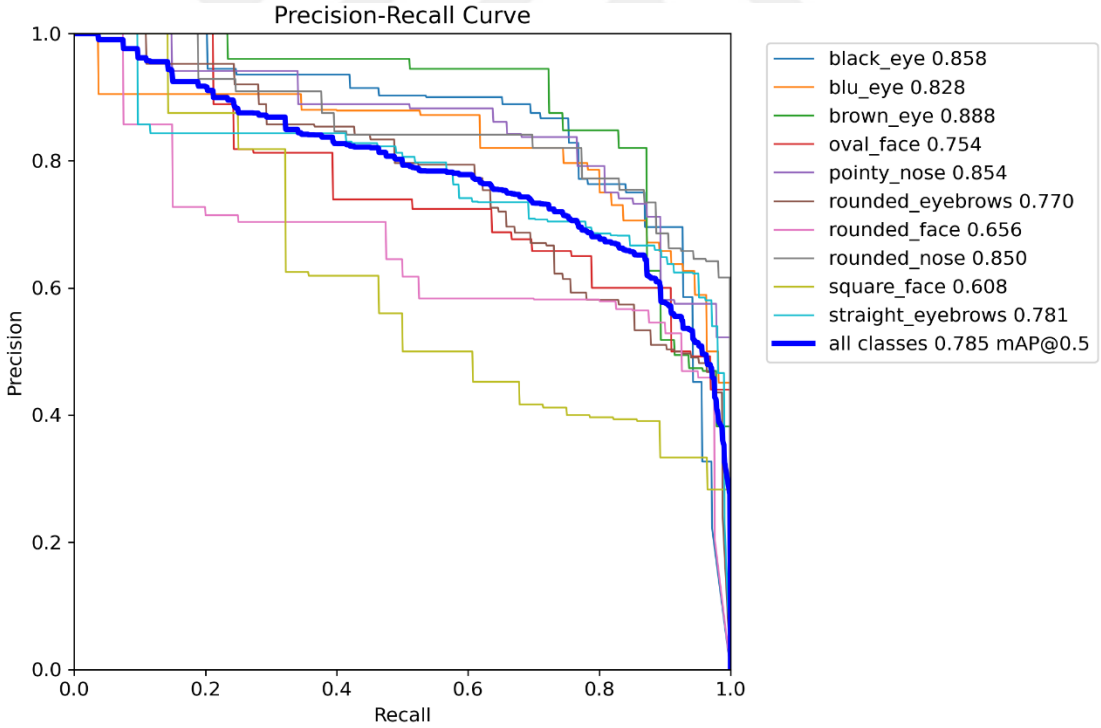

Figure 6.12. P-R-curve of all classes for trial 12.

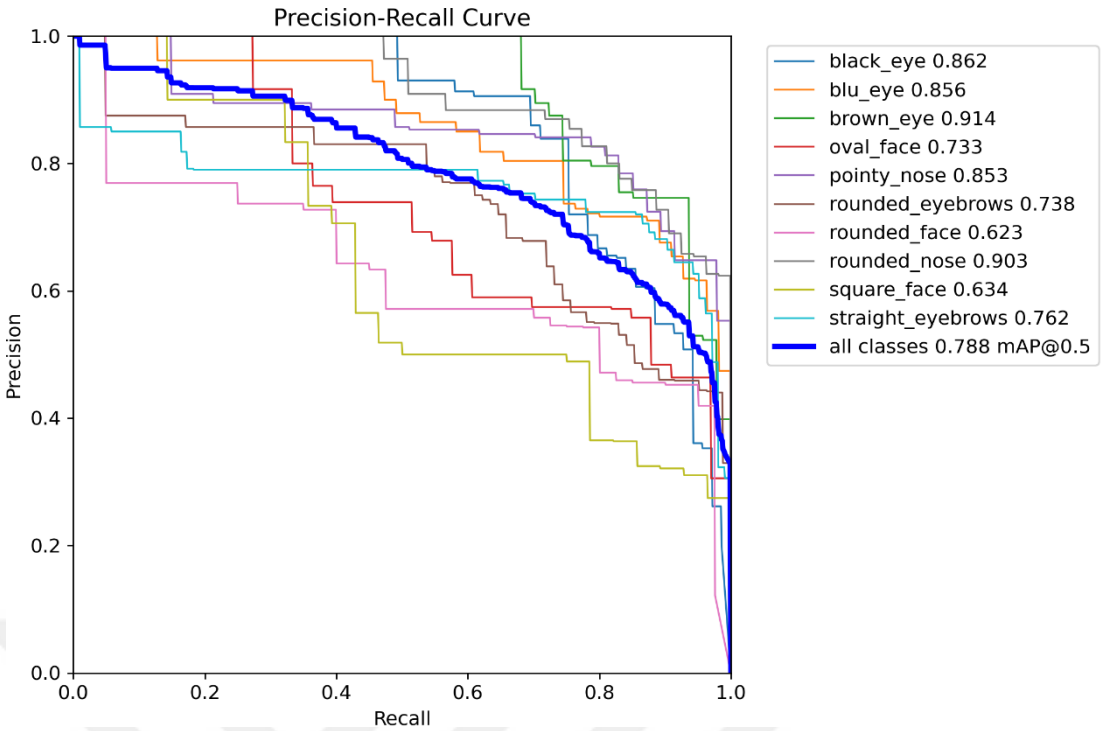

Figure 6.13. P-R-curve of all classes for trial 13

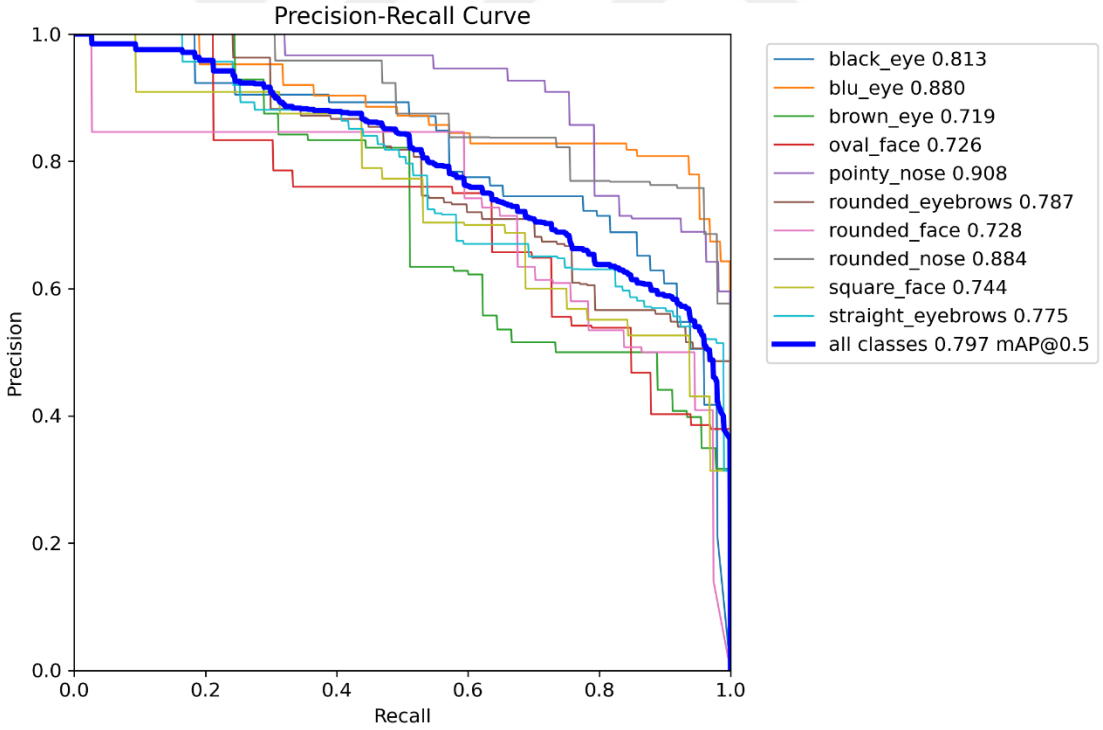

Figure 6.14. P-R-curve of all classes for trial 14.

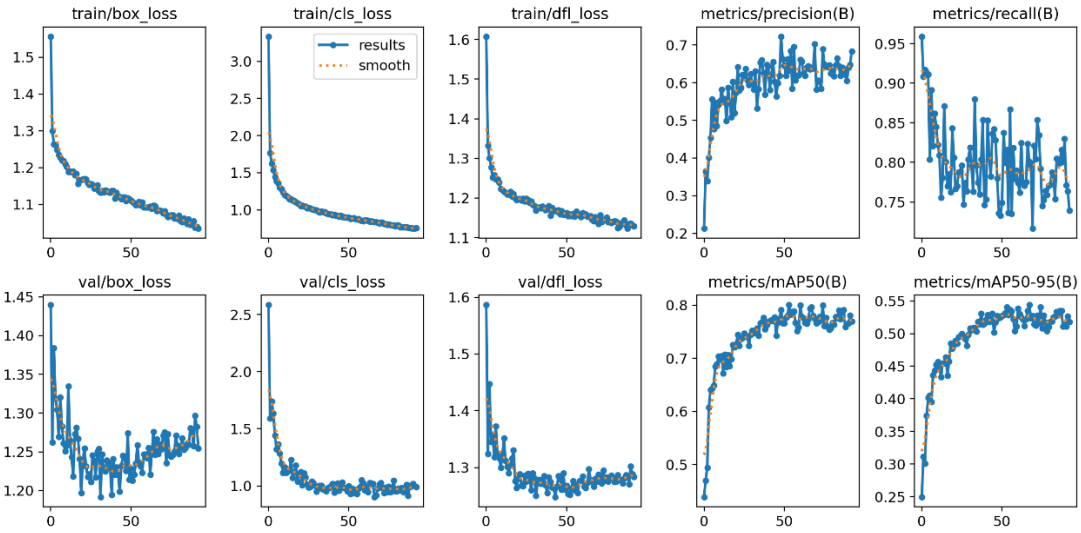

Figure 6.15. Results for training and validation for trial 14

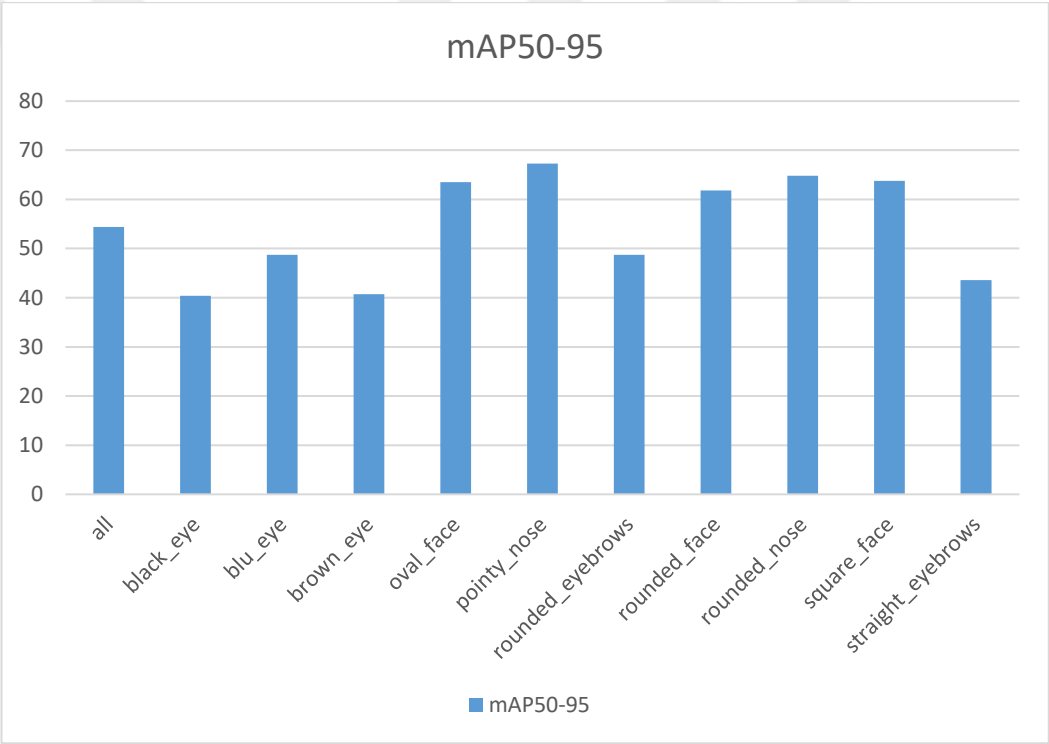

Figure 6.16. mAP (50-95) of all classes for trial 14 (Val).

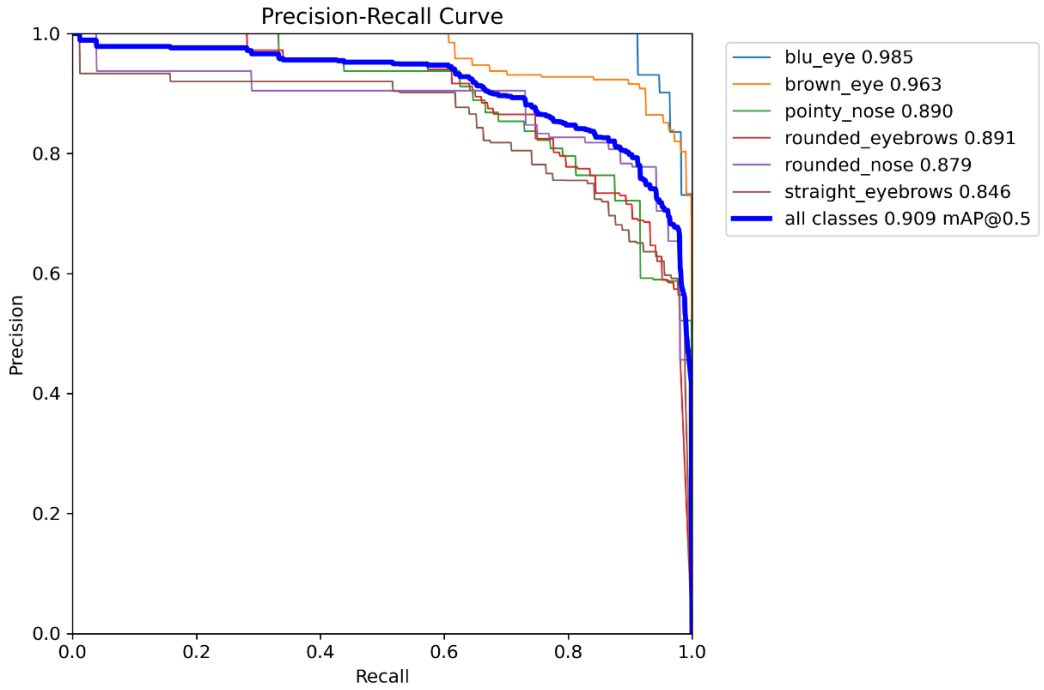

Figure 6.17. P-R-curve of all classes for trial 15

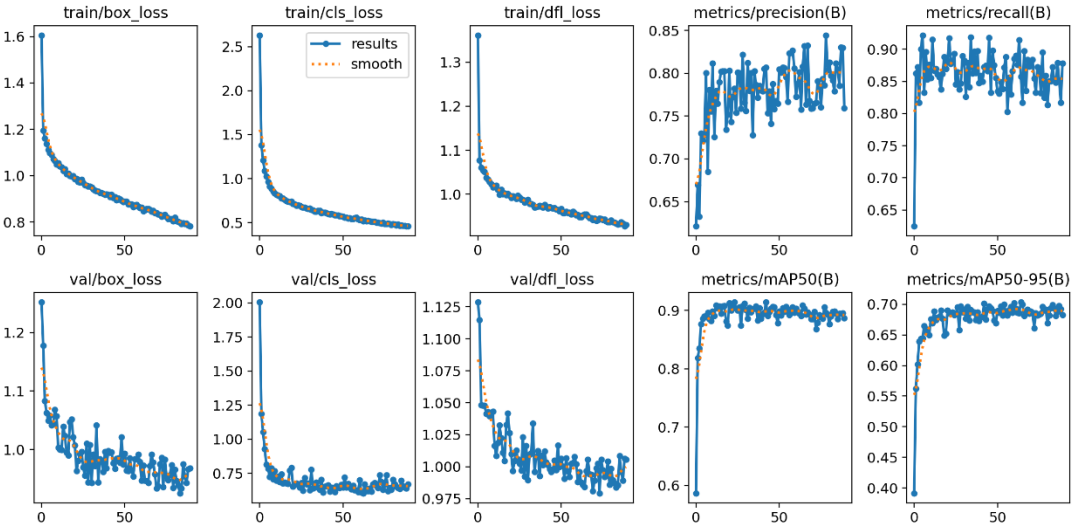

Figure 6.18. Results for training and validation for trial 15.

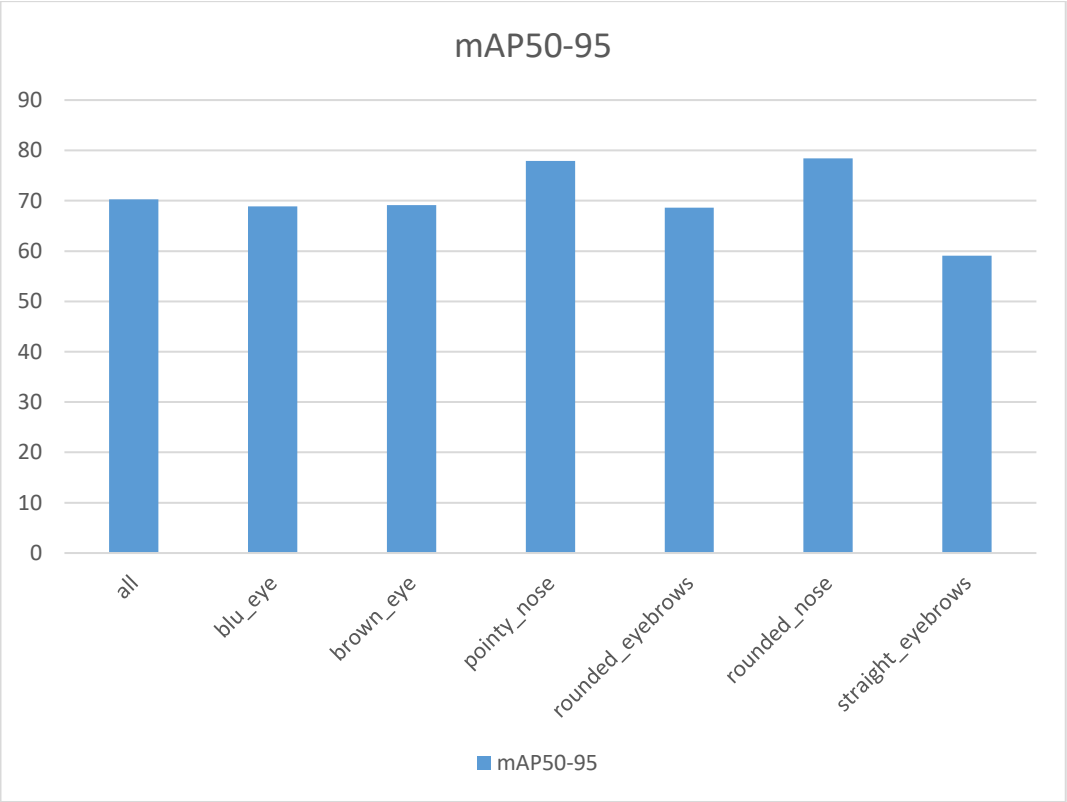

Figure 6.19. mAP (50-95) of all classes for trial 15 (Val).

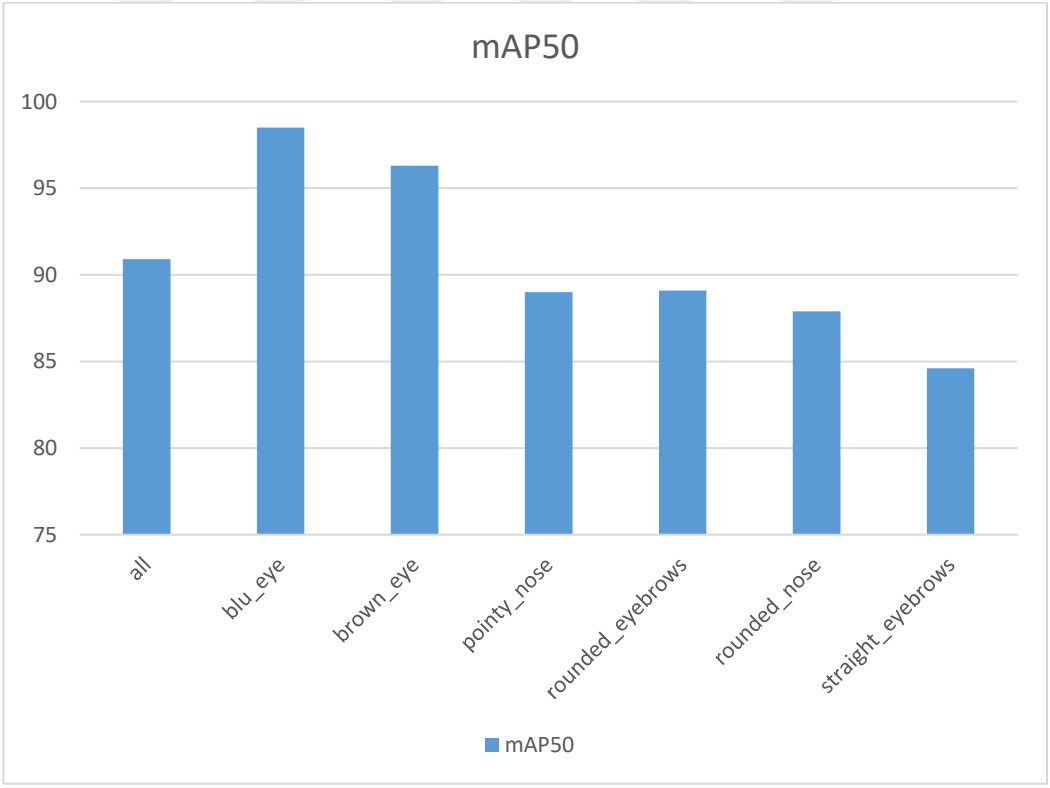

Figure 6.20. mAP (50) of all classes for trial 15 (Val).

Table 6.2 performance results for trial 15 (Val) (%)

Classes	R	P	F1	Specificity	accuracy
blu_eye	98.2	91.8	94.9	98.7	98.6
brown_eye	91.5	100	95.6	100	98
pointy_nose	68.7	82.5	75	98.2	95.1
rounded_eyebrows	81.5	80	80.7	95.1	92.5
rounded_nose	86.5	76.2	81	96.5	95.3
straight_eyebrows	74.1	80.4	77.1	96.3	92.5
Average	**83.4**	**85.1**	**84**	**97.4**	**95.3**

6.2 Model test

To evaluate the model and test its ability to locate and classify different classes, it must be tested on unseen images.

images and how the model (trial 14) locates and classifies the facial features will be presented in the upcoming figures.

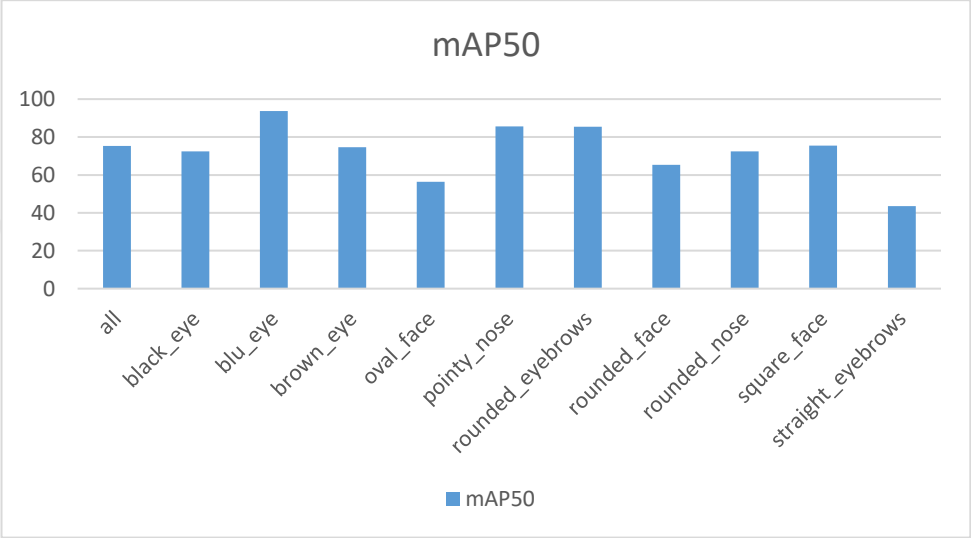

Figure 6.21. mAP-50 of all classes for trial 14 (test).

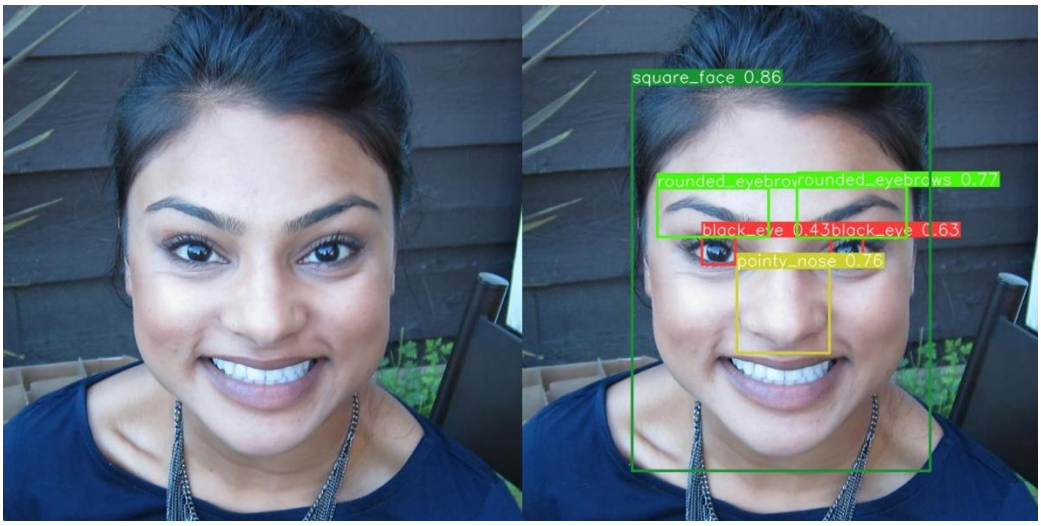

Figure 6.22. Trial 14 test sample 1.

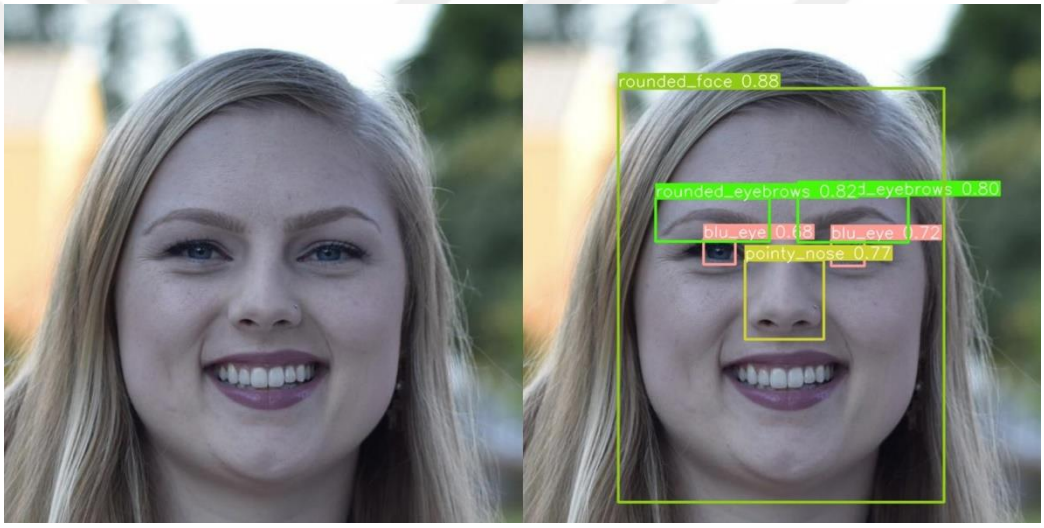

Figure 6.23. Trial 14 test sample 2.

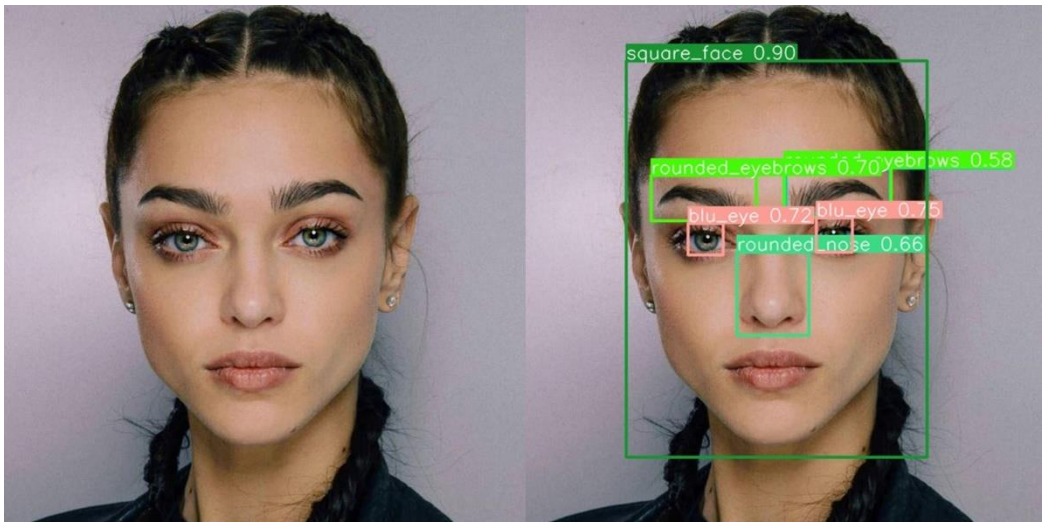

Figure 6.24. Trial 14 test sample 3.

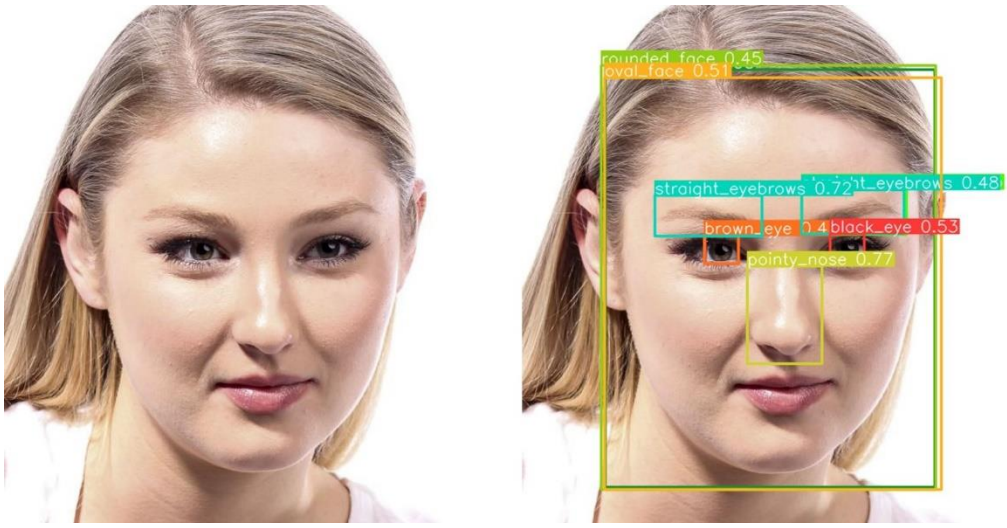

Figure 6.25. Trial 14 test sample 4.

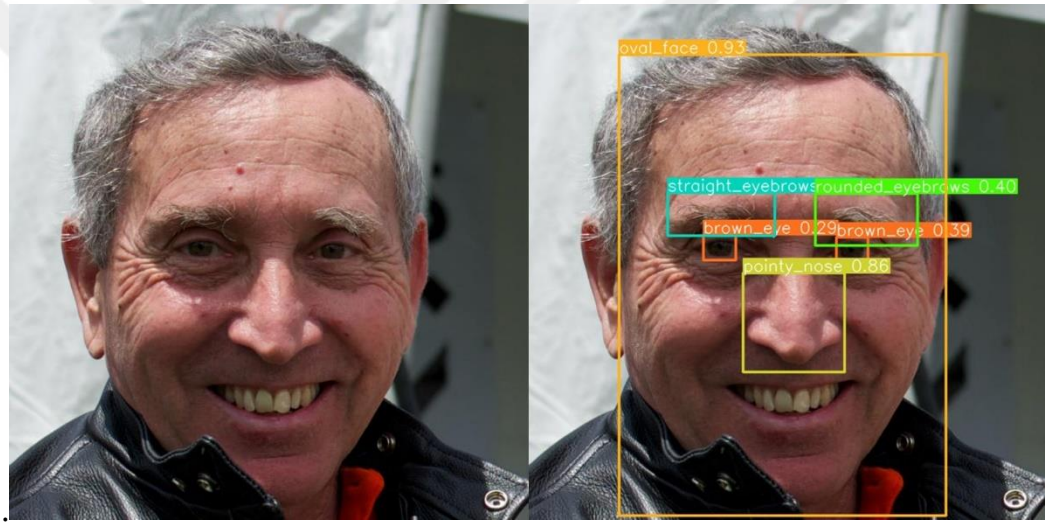

Figure 6.26. Trial 14 test sample 5.

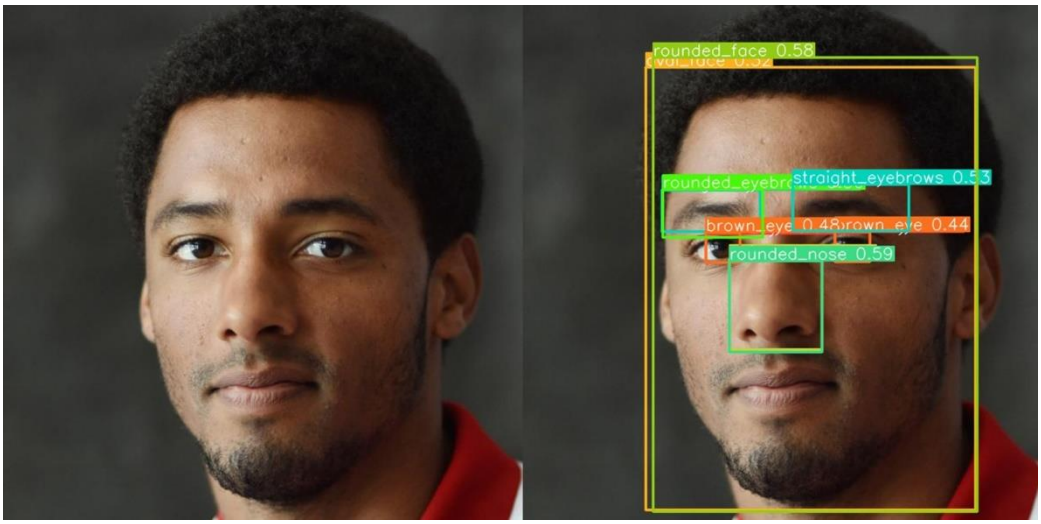

Figure 6.27. Trial 14 test sample 6.

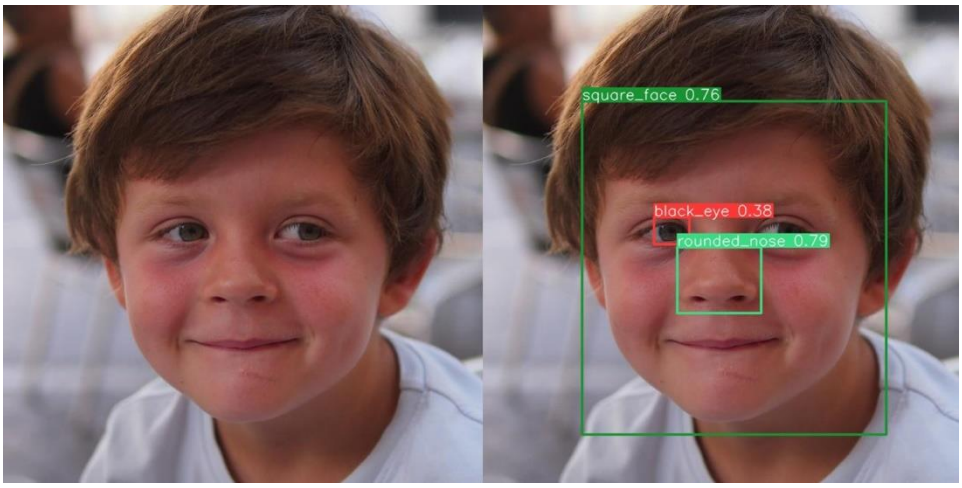

Figure 6.28. Trial 14 test sample 7.

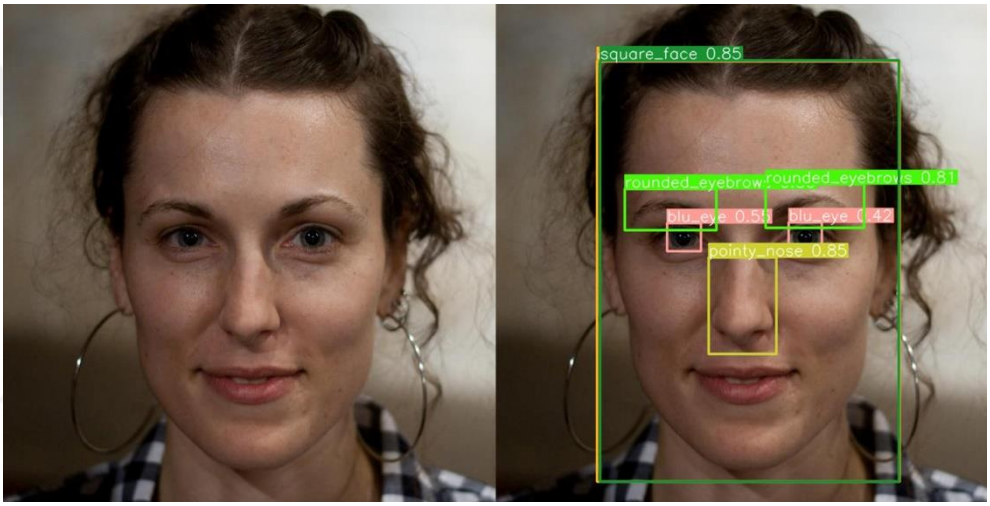

Figure 6.29. Trial 14 test sample 8.

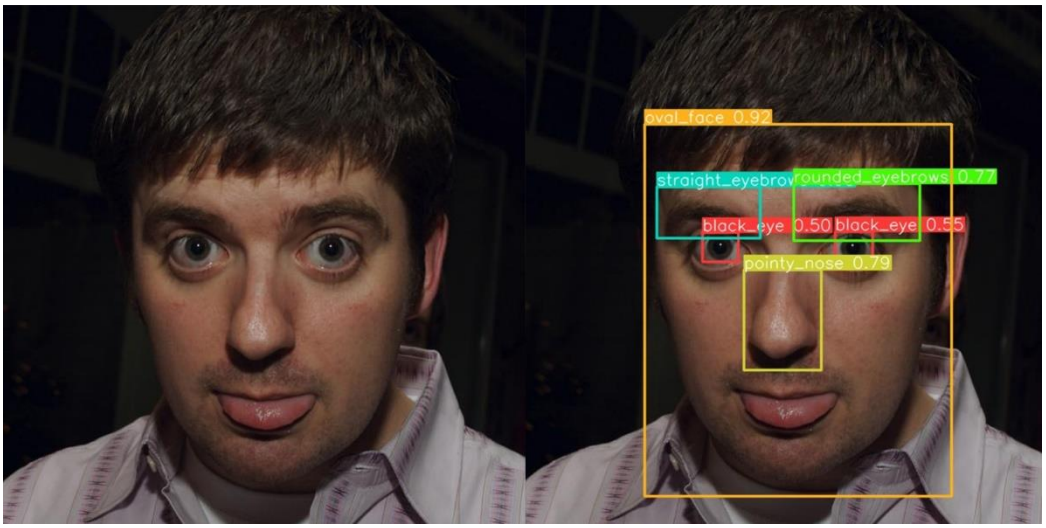

Figure 6.30. Trial 14 test sample 9.

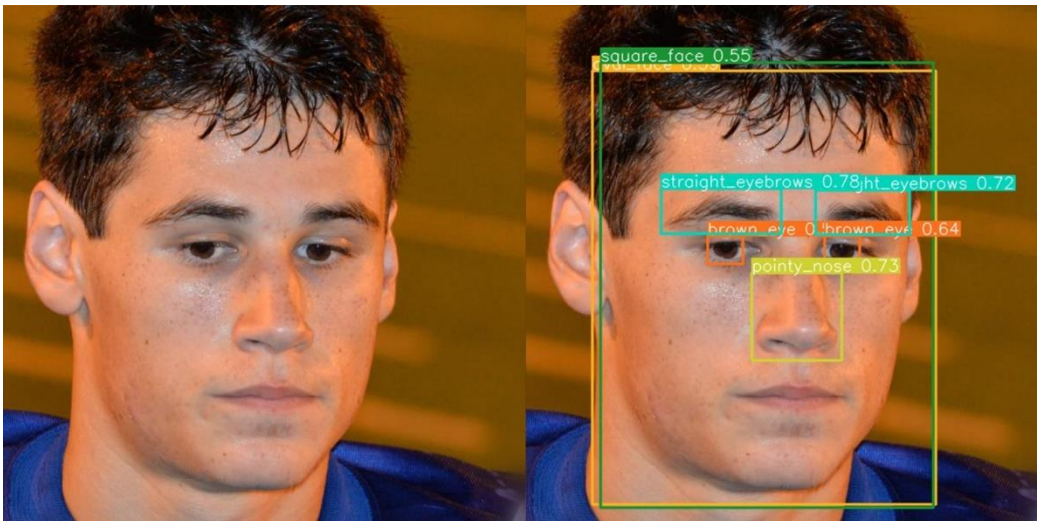

Figure 6.31. Trial 14 test sample 10.

It is noticeable that the model predictions are affected by different light conditions and face poses (samples 4, 9) the model predicts the eye as black_eye due to poor light conditions, and the face pose affected the face_shape prediction in samples 4, 6,10.
As it mentioned before a new dataset was created to handle the issue of inefficient data amount (trial 15), test result for the model will be shown on the upcoming figures.

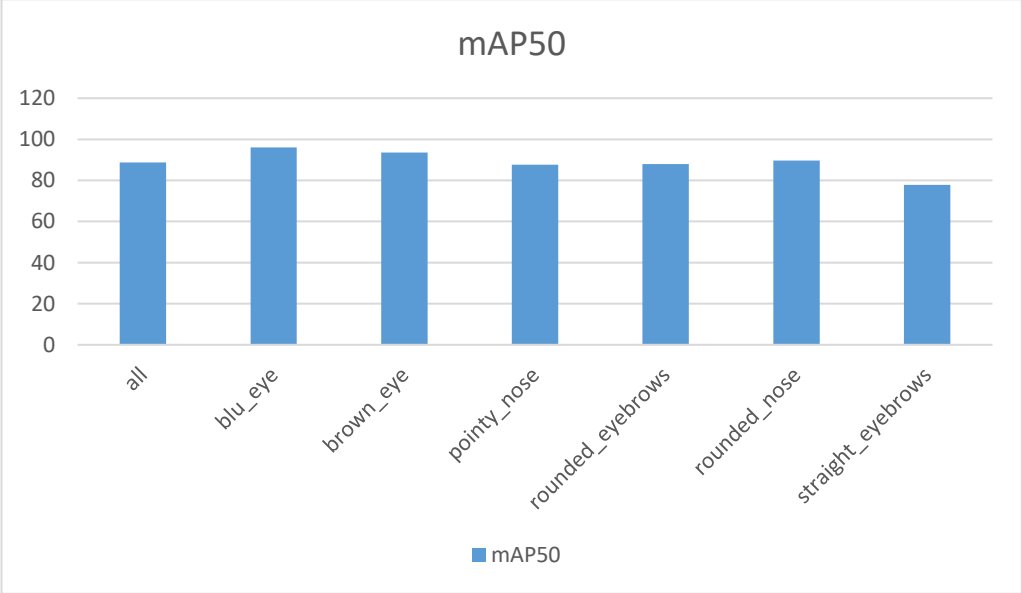

Figure 6.32. mAP (50) of all classes for trial 15 (test).

Table 6.3 performance results for trial 15(test) (%).

Classes	R	P	F1	Specificity	accuracy
blu_eye	100	100	100	100	100
brown_eye	93.1	100	96.4	100	98.7
pointy_nose	71.4	80	75.4	97.5	94.4
rounded_eyebrows	82.4	87	84.6	96	92.7
rounded_nose	80	74	76.9	96.6	94.8
straight_eyebrows	78.2	80	79.1	95.2	91.8
Average	**84.1**	**86.8**	**85.4**	**97.5**	**95.4**

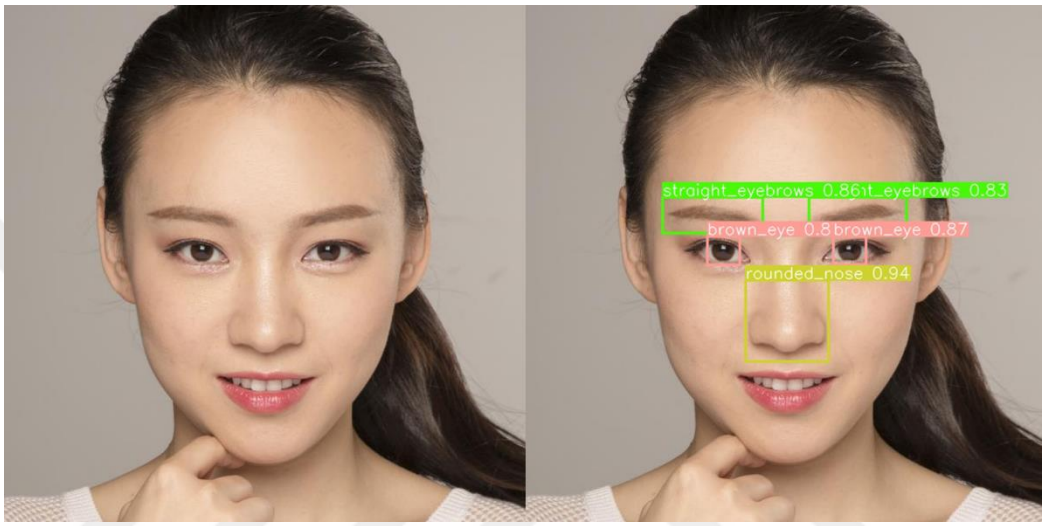

Figure 6.33. Trial 15 test sample 1.

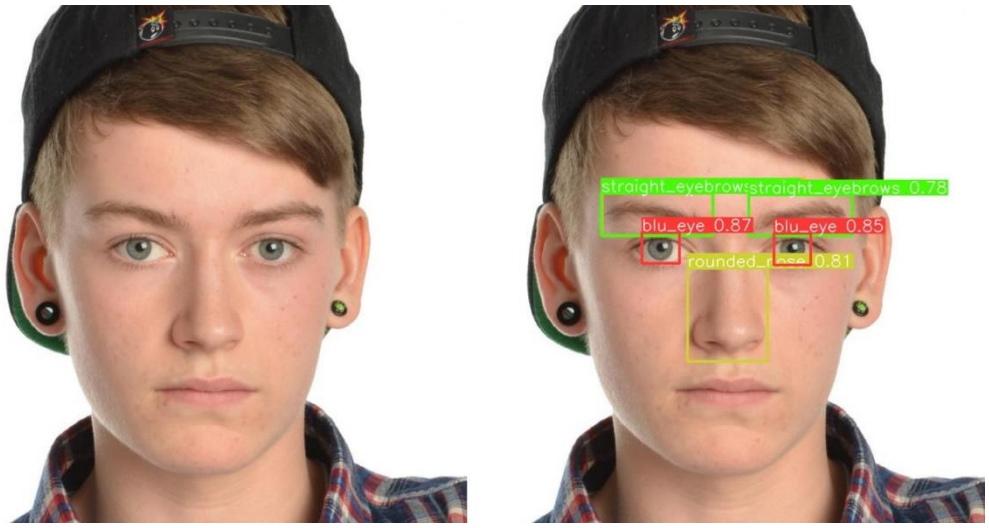

Figure 6.34. Trial 15 test sample 2.

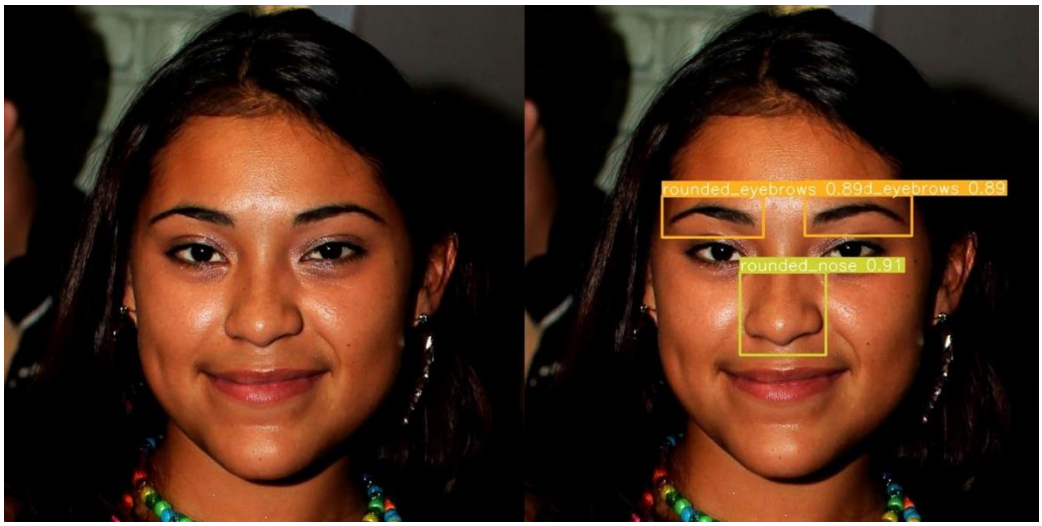

Figure 6.35. Trial 15 test sample 3.

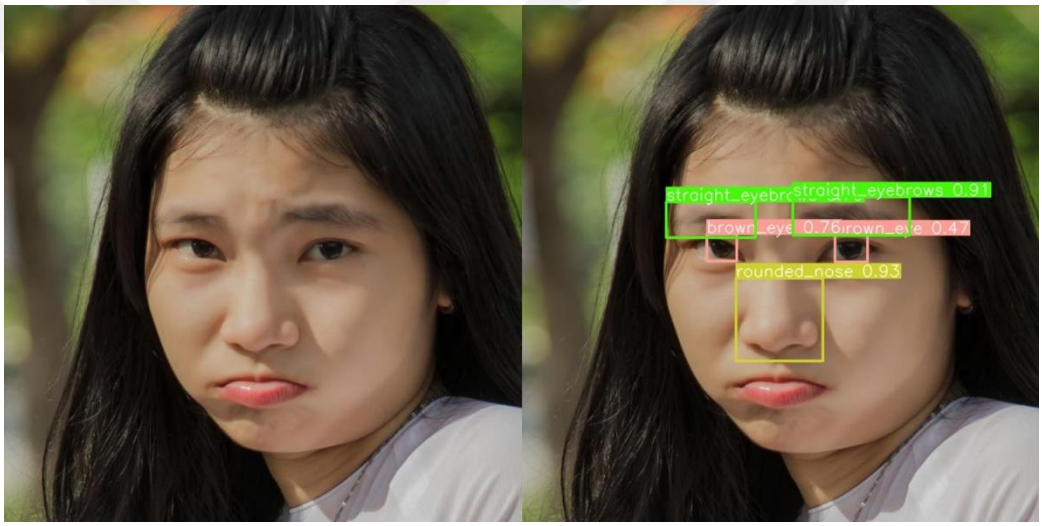

Figure 6.36. Trial 15 test sample 4.

Figure 6.37. Trial 15 test sample 5.

Figure 6.38. Trial 15 test sample 6.

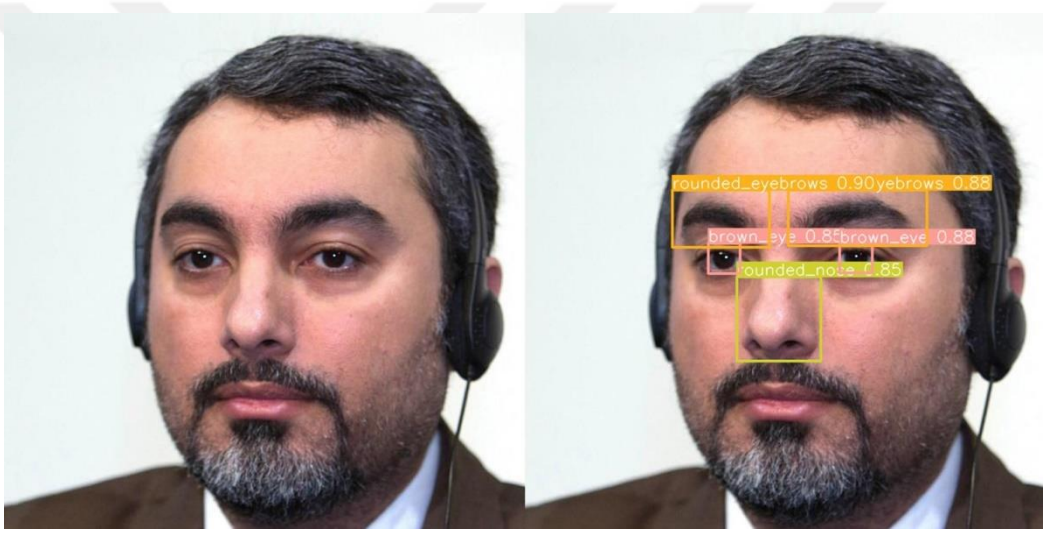

Figure 6.39. Trial 15 test sample 7.

Figure 6.40. Trial 15 test sample 8.

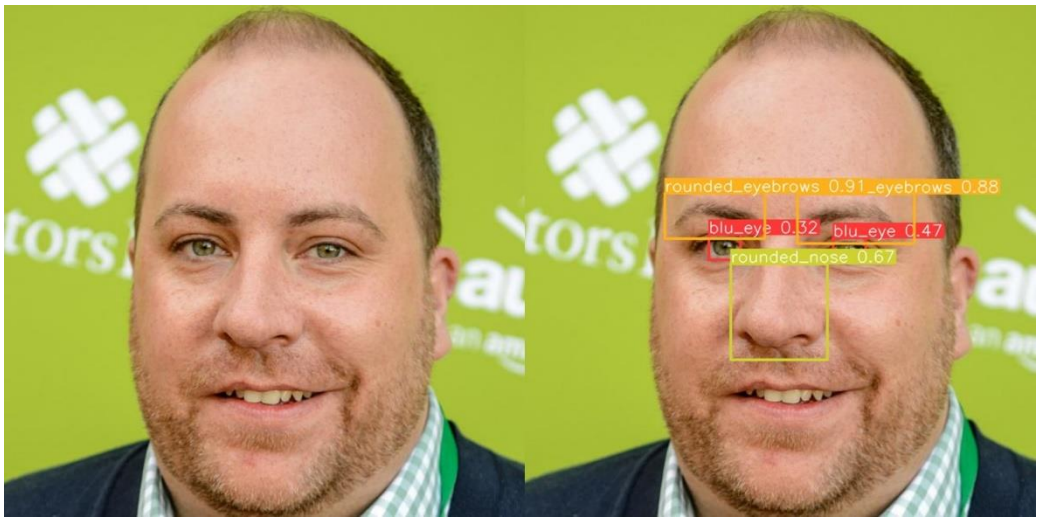

Figure 6.41. Trial 15 test sample 9.

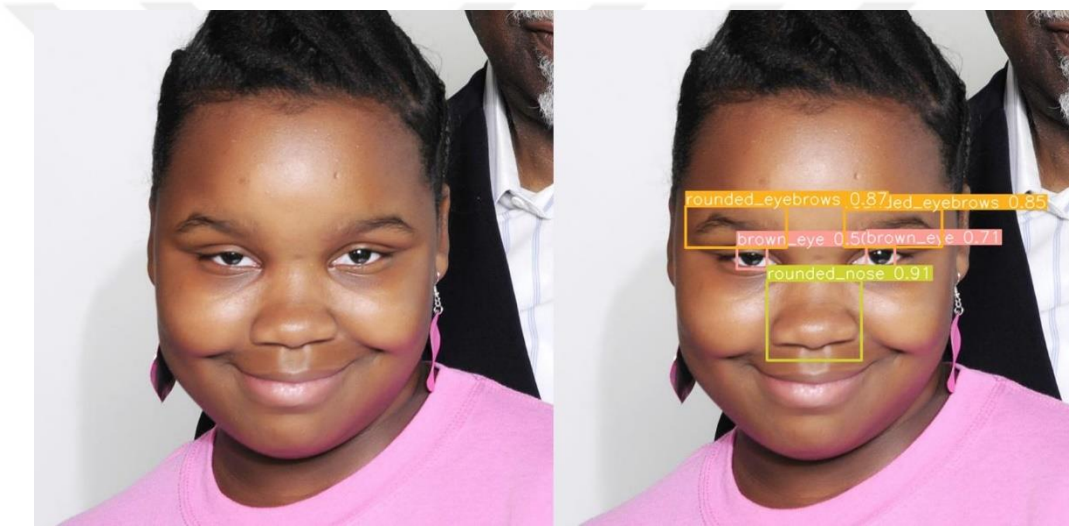

Figure 6.42. Trial 15 test sample 10.

6.2.1 Test Result Summary

Summary of test result for all trials will be presented in the Table below.

Table 6.4. Test results of all trials.

Trial	mAP ^{50}(%)	mAP $^{50-95}$(%)	Precision	Recall
1	74.1	47.6	0.560	0.873
2	73.8	48.6	0.562	0.855
3	76.9	50.4	0.621	0.778
4	70.5	45.5	0.556	0.821
5	71.4	45.9	0.550	0.859
6	70.9	46.9	0.606	0.758
7	72.2	46.3	0.541	0.849
8	73.1	47.9	0.529	0.852
9	67.1	43.5	0.520	0.853
10	71.8	46	0.554	0.840
11	72.2	47.4	0.607	0.758
12	74.6	49.1	0.557	0.877
13	75.6	50.2	0.611	0.795
14	67.7	46.1	0.581	0.659
15	**88.8**	**69.3**	**0.841**	**0.839**

6.3 Modified YOLOv8

The proposed model employed to phsiogonomy2 increases the map50 by 1.4% compared to the best version as it reaches a mAP50 of 91.3%; a detailed comparison of the proposed model with YOLOv8 variants is illustrated in the upcoming figures.

The modified YOLOv8 achieved an outstanding mAP50 of 91.3% and the highest recall rate of 87.7% compared to the previous five versions. However, there was a slight decrease of 3.1% in the precision score compared to the extra-large model. Overall, the modified version performed better than the original version, especially in the shape-related classes, which were previously a weakness.

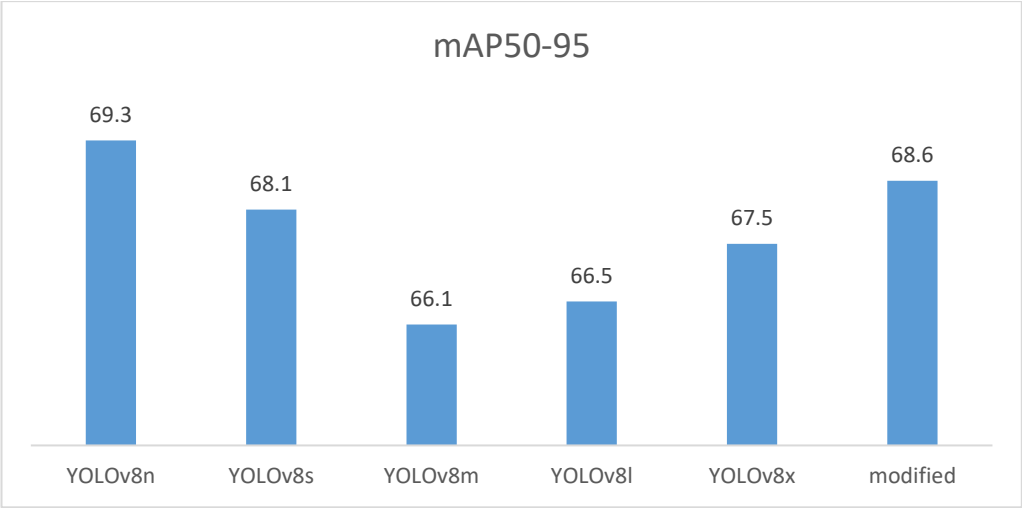

Figure 6.43. Comparison based on mean average precision 50-95.

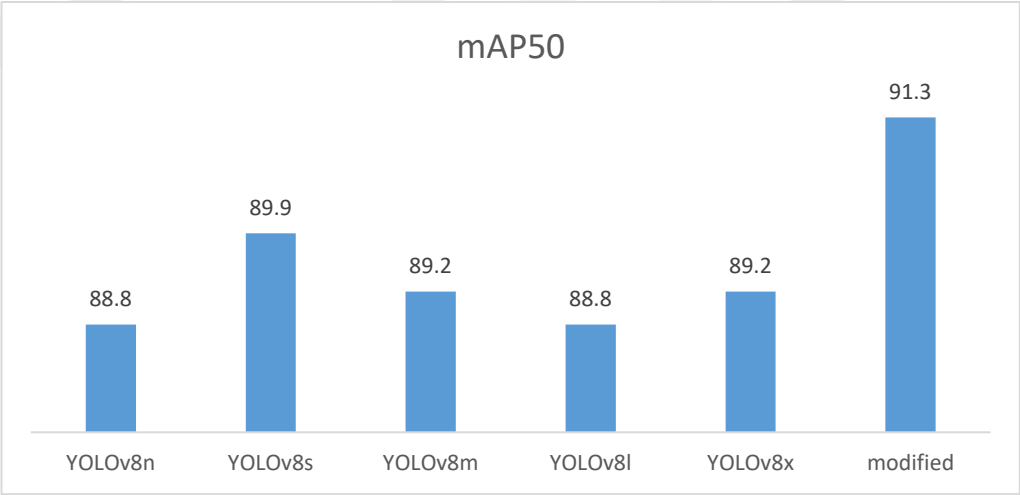

Figure 6.44. Comparison based on mean average precision 50.

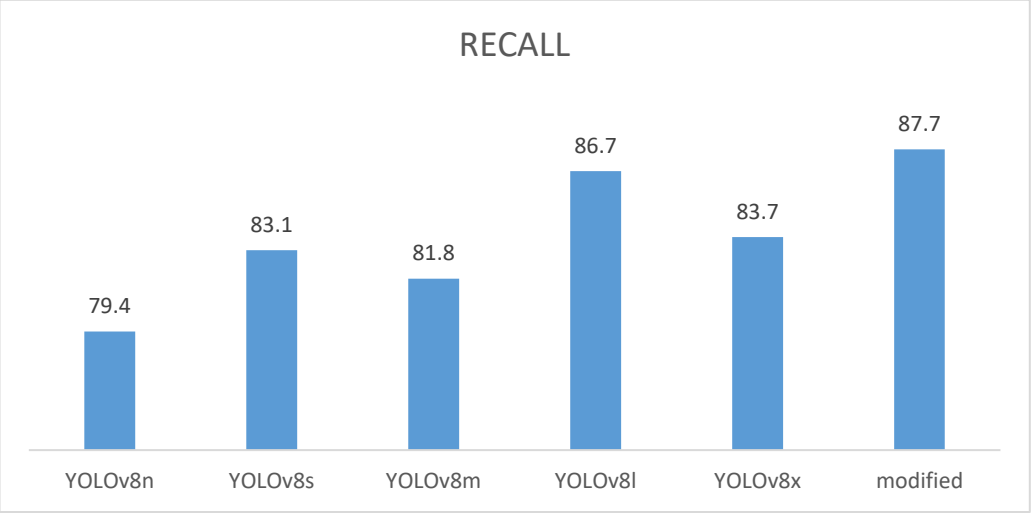

Figure 6.45. Comparison based on recall.

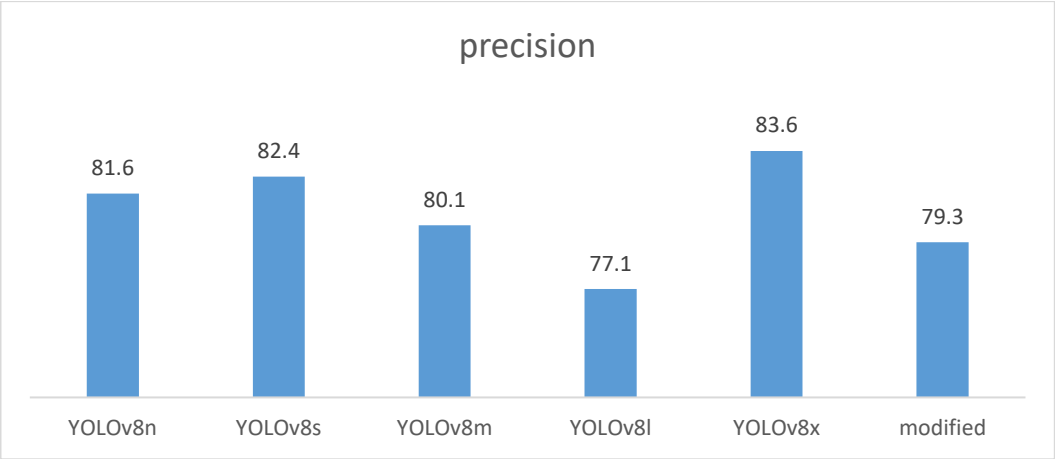

Figure 6.46. Comparison based on precision.

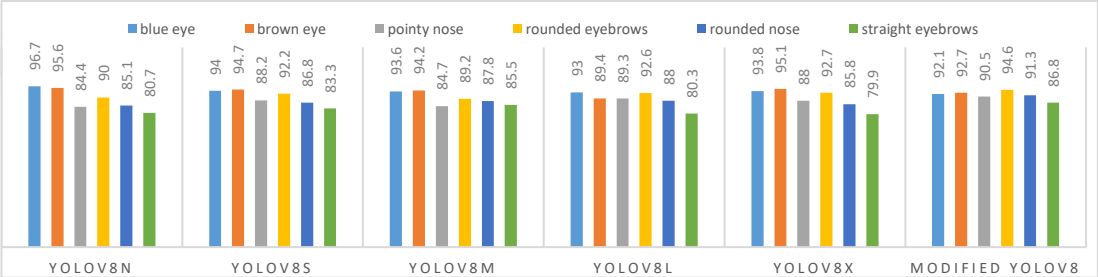

Figure 6.47. Comparison with YOLOv8 for all classes based on mAP50.

6.3.1 Compression with Related Work

Ruoling. Ma and Ruoyuan. [36] Utilized the YOLOv4 target facial expression recognition network to perform facial expression recognition. They proposed several methods to enhance the accuracy and speed of facial expression recognition and developed the PSA-YOLO target facial expression recognition network. They simplified the CSPDarknet53 backbone structure by reducing the number of residual blocks, network layers, parameters, and computation burden. Resulted in a faster recognition speed and higher accuracy. Inspired by their work, we also reduced the number of layers for the YOLOv8. Led to better accuracy, especially for shape-related classes.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this study, two carefully annotated facial feature recognition datasets were proposed. The datasets were specially created for the YOLO object detection models. The dataset was annotated carefully to ensure the highest accuracy and quality. The uniqueness of these datasets lies in their approach as it classifies facial features based on their shapes and colors, offering a new way of facial feature detection; the final dataset contains 2,150 images with Over 10K annotation with six classes, namely blue eye, brown eye, and rounded nose, rounded eyebrows, pointy nose, and straight eyebrows, our dataset serves as a valuable resource for elevating the field of computer vision in the context of facial feature recognition.

All versions of YOLOv8 were employed on both datasets; the best performance between the YOLOv8 variant on physiognomy1 was the Nano version with mAP50 of 75.6%. The small version of YOLOv8 shows a remarkable performance on physiognomy2 as it reaches mAP50 of 89.9%. Overall, the best classification result observed in color-related classes, such as the blue eye class got mAP50 of 96.7% on the nano version, and the brown eye class got mAP50 of 95.6% on the same model.

Furthermore, a modified version of YOLOv8 was proposed, reducing the depth and width of the network to 0.1 of the actual width and depth. Also, the stride of the second and third convolutional layer of the backbone is set to one to increase feature extraction; the resulting modified YOLOv8 is lightweight, has fewer layers and requires fewer computation resources, resulting in faster training time; the new model outperformed the original model as it reached an mAP of 91.3%, the modified version performed better than the original version, especially in the shape-related classes, which were previously a weakness, increasing the mAP50 of pointy nose, rounded eyebrows, rounded nose, straight eyebrows, by 1.2%, 1.9%, 3.3%, 1.3%, respectively.

7.2 Future Work

The work in this study can be further enhanced, to get a better result there are several improvements that can be done:

1. Adding more data will help the model classify different classes more accurately.
2. Adding more classes for different features shapes and colors.
3. Adding data with different light conditions, resolutions, face poses, and different distances between the camera and the face will help the model to be more generalized.

REFERENCES

[1] D. McNeill. The Face: A Natural History, New York, USA: Back Bay Books, 2000.

[2] A. Todorov, C. P. Said, A. D. Engell, N. N. Oosterhof. Understanding evaluation of faces on social dimensions. Trends in Cognitive Sciences, vol. 12, no. 12, pp. 455–460, 2008.

[3] C. C. Ballew II, A. Todorov. Predicting political elections from rapid and unreflective face judgments. Proceedings of T. Zhang et al. / Physiognomy: Personality Traits Prediction by Learning the National Academy of Sciences of the United States of America, vol. 104, no. 46, pp. 17948–17953, 2007.

[4] I. V. Blair, C. M. Judd, K. M. Chapleau. The influence of afrocentric facial features in criminal sentencing. Psychological Science, vol. 15, no. 10, pp. 674–679, 2004.

[5] Willis, J., & Todorov, A. (2006). First impressions: Making up your mind after a 100-ms exposure to a face. *Psychological Science*, *17*(7), 592–598. https://doi.org/10.1111/J.1467-9280.2006.01750.X

[6] How your looks betray your personality | New Scientist. (n.d.). Retrieved April 29, 2023, from https://www.newscientist.com/article/mg20126957-300-how-your-looks-betray-your-personality/

[7] R. Szeliski, "Introduction" in Computer vision: algorithms and applications, R.Szeliski, London: Springer, 2011, pp. 1-10.

[8] el Naqa Issam and Murphy, M. J. (2015). What Is Machine Learning? In R. and M. M. J. el Naqa Issam and Li (Ed.), *Machine Learning in Radiation Oncology: Theory and Applications* (pp. 3–11). Springer International Publishing. https://doi.org/10.1007/978-3-319-18305-3_1

[9] A. L. Samuel, "Some studies in machine learning using the game of checkers,"IBM Journal of Research and Development, vol. 3, no. 3, pp. 210-229, July 19

[10] E. R. Davies, “Statistical Pattern Recognition”, in Computer and MachineVision: Theory, Algorithms, Practicalities”, 4th ed. Waltham: Elsevier, 2012, pp.681.

[11] R. Dechter. “Learning while searching in constraint-satisfaction-problems,” 5th AAAI National Conference on Artificial Intelligence, Calabasas, CA, 1986, pp. 178-183.

[12] N. N. Aizenberg, I. Aizenberg and J. P.L. Vandewalle, Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications, Berlin: Springer, 2000.

[13] W. J. Zhang, G. Yang, Y. Lin, C. Ji and M. M. Gupta, "On Definition of Deep Learning," *2018 World Automation Congress (WAC)*, Stevenson, WA, USA, 2018, pp. 1-5, doi: 10.23919/WAC.2018.8430387.

[14] Y. LeCun, Y. Bengio and G. Hinton, “Deep Learning,” Nature, vol. 521, pp.436-444, May 2015.

[15] Ian Goodfellow, Yoshua Bengio, & Aaron Courville. *DeepLearning* .

[16] Zeiler, M. D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53.

[17] Eluyode, O., Applied, D. A.-E. J. of, & 2013, undefined. (n.d.). Comparative study of biological and artificial neural networks.

[18] Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, *22*(5), 717–727. https://doi.org/https://doi.org/10.1016/S0731-7085(99)00272-1.

[19] W.S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” Bulletin of Mathematical Biophysics, vol.5, pp. 115-133, Dec1943.

[20] IMRAN AHMED. (2020). *40 ALGORITHMS EVERY PROGRAMMER SHOULD KNOW* (Commissioning Editor: Kunal Chaudhari, Acquisition Editor: Karan Gupta, Content Development Editor: Pathikrit Roy, Senior Editor: Rohit

Singh, & Technical Editor: Pradeep Sahu, Eds.; 1st ed.). PACKT PUBLISHING LIMITED.

[21] M. Elgendy “Deep learning and neural networks”, in Deep Learning for Vision Systems, 6th ed. New Work: Manning, 2019, pp. 36-41.

[22] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” Biol. Cybernetics, vol. 36, pp. 193-202, Apr 1980.

[23] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MA:MIT press,2016.

[24] S. Khan, H. Rahmani, S. A. A. Shah and M. Bennamoun, “Convolutional neural network”, in A Guide to Convolutional Neural Networks for Computer Vision, 1st ed. California: Morgan & Claypool, 2018, pp.45-56.

[25] R. Chellappa, C. L. Wilson, and S. Sirohey, “Human and machine recognition of faces: A survey,” Proc. IEEE, vol. 83, no. 5, pp. 705–740, May 1995.

[26] Zuo W, Zhang D, Yang J, Wang K. BDPCA plus LDA: a novel fast feature extraction technique for face recognition. IEEE Trans Syst Man Cybern B Cybern. 2006 Aug;36(4):946-53. doi: 10.1109/tsmcb.2005.863377. PMID: 16903378.

[27] R. Chellappa, C. L. Wilson and S. Sirohey, "Human and machine recognition of faces: a survey," in *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705-741, May 1995, doi: 10.1109/5.381842.

[28] Wen Yandong and Zhang, K. and L. Z. and Q. Y. (2016). A Discriminative Feature Learning Approach for Deep Face Recognition. In J. and S. N. and W. M. Leibe Bastian and Matas (Ed.), *Computer Vision – ECCV 2016* (pp. 499–515). Springer International Publishing.

[29] Handbook of Face Recognition. (2011). In Stan Z. Li & Anil K. Jain (Eds.), *Handbook of Face Recognition*. Springer London. https://doi.org/10.1007/978-0-85729-932-1.

[30] Hechun, W., & Xiaohong, Z. (2019). Survey of Deep Learning Based Object Detection. *Proceedings of the 2nd International Conference on Big Data Technologies*, 149–153. https://doi.org/10.1145/3358528.3358574.

[31] Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R. (2019). A Survey of Deep Learning-Based Object Detection. *IEEE Access*, *7*, 128837–128868. https://doi.org/10.1109/ACCESS.2019.2939201.

[32] S. Ren, K. He, R. Girshick, and J. Sun, ‘‘Faster R-CNN: Towards real-timeobject detection with region proposal networks,’’IEEE Trans. PatternAnal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once:Unified, real-time object detection,’’ inProc. IEEE Conf. Comput. Vis.Pattern Recognit., Jun. 2016, pp. 779–788.

[34] Liu Wei and Anguelov, D. and E. D. and S. C. and R. S. and F. C.-Y. and B. A. C. (2016). SSD: Single Shot MultiBox Detector. In J. and S. N. and W. M. Leibe Bastian and Matas (Ed.), *Computer Vision – ECCV 2016* (pp. 21–37). Springer International Publishing.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, ‘‘Mask R-CNN,’’ inProc.IEEE ICCV, Dec. 2017, pp. 2980–2988.

[36] Druzhkov, P. N., & Kustikova, V. D. (2016). A survey of deep learning methods and software tools for image classification and object detection. Pattern Recognition and Image Analysis, 26(1), 9–15. https://doi.org/10.1134/S1054661816010065.

[37] S. Z. Li and Juwei Lu, "Face recognition using the nearest feature line method," in IEEE Transactions on Neural Networks, vol. 10, no. 2, pp. 439-443, March 1999, doi: 10.1109/72.750575.

[38] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1, I–I. https://doi.org/10.1109/CVPR.2001.990517.

[39] D. Nguyen, D. Halupka, P. Aarabi, and A. Sheikholeslami, "Real-time face detection and lip feature extraction using field-programmable gate arrays," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 36, no. 4, pp. 902-912, Aug. 2006, doi: 10.1109/TSMCB,2005,862728.

[40] M. P. Pamplona Segundo, L. Silva, O. R. P. Bellon and C. C. Queirolo, "Automatic Face Segmentation and Facial Landmark Detection in Range Images,"

in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 40, no. 5, pp. 1319-1330, Oct. 2010, doi: 10.1109/TSMCB.2009.2038233.

[41] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, "Real-time facial feature detection using conditional regression forests," 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 2012, pp. 2578-2585, doi: 10.1109/CVPR.2012.6247976.

[42] Tresadern, P.A., Ionita, M.C. & Cootes, T.F. Real-Time Facial Feature Tracking on a Mobile Device. Int J Comput Vis 96, 280–289 (2012). https://doi.org/10.1007/s11263-011-0464-9.

[43] Zhenyao Zhu, Ping Luo, Xiaogang Wang, & Xiaoou Tang. (2013). Deep Learning Identity-Preserving Face Space. International Conference on Computer Vision (ICCV).

[44] Y. Sun, X. Wang and X. Tang, "Deep Learning Face Representation from Predicting 10,000 Classes," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 1891-1898, doi: 10.1109/CVPR.2014.244.

[45] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. IEEE Signal Processing Letters, 23(10), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342.

[46] S. Guo, S. Chen and Y. Li, "Face recognition based on convolutional neural network and support vector machine," 2016 IEEE International Conference on Information and Automation (ICIA), Ningbo, China, 2016, pp. 1787-1792, doi:10.1109/ICInfA.2016.7832107.

[47] Muhammad Asim, Zhu Ming and M. Y. Javed, "CNN based spatiotemporal feature extraction for face anti-spoofing," 2017 2nd International Conference on Image, Vision and Computing (ICIVC), Chengdu, 2017, pp. 234-238, doi: 10.1109/ICIVC.2017.7984552.

[48] Zhang, T., Qin, R.-Z., Dong, Q.-L., Gao, W., Xu, H.-R., & Hu, Z.-Y. (2017). Physiognomy: Personality traits prediction by learning. International Journal of Automation and Computing, 14(4), 386–395. https://doi.org/10.1007/s11633-017-1085-8.

[49] Li, J., Zhang, D., Zhang, J., Zhang, J., Li, T., Xia, Y., Yan, Q., & Xun, L. (2017). Facial Expression Recognition with Faster R-CNN. Procedia Computer

Science, 107, 135–140. https://doi.org/https://doi.org/10.1016/j.procs.2017.03.069.

[50] X. Ziwei et al., "Face Occlusion Detection Based on SSD Algorithm," 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 2020, pp. 362-365, doi: 10.1109/ICEIEC49280.2020.9152335.

[51] Kumar, A., Kalia, A., Verma, K., Sharma, A., & Kaushal, M. (2021). Scaling up face masks detection with YOLO on a novel dataset. Optik, 239, 166744.

[52] Nguyen Quoc, H., & Truong Hoang, V. (2021). Real-Time Human Ear Detection Based on the Joint of Yolo and RetinaFace. Complexity, 2021, 7918165. https://doi.org/10.1155/2021/7918165.

[53] Chen, W., Huang, H., Peng, S., Zhou, C., & Zhang, C. (2021). YOLO-face: a real-time face detector. The Visual Computer, 37(4), 805–813. https://doi.org/10.1007/s00371-020-01831-7.

[54] Bhagat, P. K., & Choudhary, P. (2018). Image annotation: Then and now, *Image and Vision Computing*, *80*, 1–23. https://doi.org/https://doi.org/10.1016/j.imavis.2018.09.017.

[55] Tero Karras, et al. "A Style-Based Generator Architecture for Generative Adversarial Networks", 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2019): 4396-4405.

[56] https://github.com/heartexlabs/labelImg

[57] https://www.roboflow.com

[58] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, *128*(2), 261–318. https://doi.org/10.1007/s11263-019-01247-4.

[59] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *ArXiv*. /abs/1506.02640.

[60] Rosetree, R. (2013). *The new power of face reading*. Women's Intuition Worldwide.

[61] Joey Yap. (2005). *Mian Xiang-Discover Face Reading* (first edition). JY Books Sdn.

[62] Chi An Kuei. (2000). *Face Reading: Keys to Instant Character Analysis*. M. Evans & Company.

[63] Farooq, dhufr. (2023, June). *Physiognomy Dataset*. Https://Universe.Roboflow.Com/Dhufr-Farooq/Physiognomy.

[64] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in'Real-Life'Images: detection, alignment, and recognition.

[65] Liu, Z., Luo, P., Wang, X., & Tang, X. (2018). Large-scale celebfaces attributes (celeba) dataset. Retrieved August, 15(2018), 11.

[66] Terhörst, P., Fährmann, D., Kolf, J. N., Damer, N., Kirchbuchner, F., & Kuijper, A. (2021). Maad-face: A massively annotated attribute dataset for face images. IEEE Transactions on Information Forensics and Security, 16, 3942-3957.

[67] Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

[68] Terven, J. R., & Cordova-Esparaza, D. M. (2023). A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. https://arxiv.org/abs/2304.00501v1

[69] lutz roeder. NETRON. www.netron.app.

[70] RangeKing. A brief summary of YOLOv8 model structure. https://github.com/ultralytics/ultralytics/issues/189

[71] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deeplearning," arXiv, pp. 1–28, 2016.

[72] Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding Batch Normalization. Advances in Neural Information Processing Systems, 31.

[73] Singh, B., Patel, S., Vijayvargiya, A., & Kumar, R. (2023). Analyzing the impact of activation functions on the performance of the data-driven gait model. Results in Engineering, 18, 101029. https://doi.org/https://doi.org/10.1016/j.rineng.2023.101029

[74] Fu, R., Li, B., Gao, Y., & Wang, P. (2018). CNN with coarse-to-fine layer for hierarchical classification. IET Computer Vision, 12(6), 892–899. https://doi.org/https://doi.org/10.1049/iet-cvi.2017.0636

[75] He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. https://doi.org/10.1007/978-3-319-10578-9_23

[76] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. http://arxiv.org/abs/1506.02640

[77] Janocha, K., & Czarnecki, W. M. (2016). On loss functions for deep neural networks in classification. Schedae Informaticae, 25, 49–59. https://doi.org/10.4467/20838476SI.16.004.6185.

[78] Lai, S.-H., Lepetit, V., Nishino, K., & Sato, Y. (Eds.). (2017). Computer Vision – ACCV 2016 (Vol. 10115). Springer International Publishing. https://doi.org/10.1007/978-3-319-54193-8.

[79] Md. R. Karim, M. Sewak and P. Pujari, "Model Performance Optimization", in Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python, 1st ed. Birmingham: Packt Publishing, 2018, pp. 83+.

[80] Brownlee, Jason. "What is the Difference Between a Batch and an Epoch in a Neural Network." Machine Learning Mastery 20 (2018).

[81] Ketkar, N. (2017). Stochastic Gradient Descent. In Deep Learning with Python (pp. 113–132). Apress.

[82] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. http://arxiv.org/abs/1409.1556.

[83] Najah, A., Mustafa, F. F., & Hacham, W. S. (2021). Building a High Accuracy Transfer Learning-Based Quality Inspection System at Low Costs. Al-Khwarizmi Engineering Journal, 17(1), 1–12.

[84] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big Data, 3(1), 9. https://doi.org/10.1186/s40537-016-0043-6

[85] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778. https://doi.org/10.1109/CVPR.2016.90

[86] Erhan, D., Bengio, Y., Courville, A., Ca, P.-A. M., Ca, P. V., & Com, B. (2010). Why Does Unsupervised Pre-training Help Deep Learning? Pierre-Antoine Manzagol Pascal Vincent Samy Bengio. In Journal of Machine Learning Research (Vol. 11).

[87] Tsung-Yi Lin, Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., … Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *CoRR*, *abs/1405.0312*. Retrieved from http://arxiv.org/abs/1405.0312.

[88] Aboah, A., Wang, B., & Bagci, U. (2023). Real-time Multi-Class Helmet Violation Detection Using Few-Shot Data Sampling Technique and YOLOv8.

APPENDIX A
Dataset yaml file

```
names: #classes names
- black_eye
- blu_eye
- brown_eye
- oval_face
- pointy_nose
- rounded_eyebrows
- rounded_face
- rounded_nose
- square_face
- straight_eyebrows
nc: 10 #number of classes
test: dataset/test/images # path to test images
train: dataset/train/images # path to train images
val: dataset/valid/images # path to validation images
```

APPENDIX B
YOLOv8 configuration file

```
# Ultralytics YOLO 🚀, AGPL-3.0 license

# YOLOv8 object detection model with P3-P5 outputs. For Usage examples see https://docs.ultralytics.com/tasks/detect

# Parameters

nc: 80  # number of classes

scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml' will call yolov8.yaml with scale 'n'

  # [depth, width, max_channels]

  n: [0.33, 0.25, 1024]  # YOLOv8n summary: 225 layers,  3157200 parameters, 3157184 gradients,   8.9 GFLOPs

  s: [0.33, 0.50, 1024]  # YOLOv8s summary: 225 layers, 11166560 parameters, 11166544 gradients,  28.8 GFLOPs

  m: [0.67, 0.75, 768]  # YOLOv8m summary: 295 layers, 25902640 parameters, 25902624 gradients,  79.3 GFLOPs

  l: [1.00, 1.00, 512]  # YOLOv8l summary: 365 layers, 43691520 parameters, 43691504 gradients, 165.7 GFLOPs

  x: [1.00, 1.25, 512]  # YOLOv8x summary: 365 layers, 68229648 parameters, 68229632 gradients, 258.5 GFLOPs

# YOLOv8.0n backbone

backbone:

  # [from, repeats, module, args]

  - [-1, 1, Conv, [64, 3, 2]]  # 0-P1/2

  - [-1, 1, Conv, [128, 3, 2]]  # 1-P2/4

  - [-1, 3, C2f, [128, True]]

  - [-1, 1, Conv, [256, 3, 2]]  # 3-P3/8

  - [-1, 6, C2f, [256, True]]
```

```
  - [-1, 1, Conv, [512, 3, 2]]  # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]]  # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]]  # 9

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 6], 1, Concat, [1]]  # cat backbone P4
  - [-1, 3, C2f, [512]]  # 12

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 4], 1, Concat, [1]]  # cat backbone P3
  - [-1, 3, C2f, [256]]  # 15 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]]  # cat head P4
  - [-1, 3, C2f, [512]]  # 18 (P4/16-medium)
  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]]  # cat head P5
  - [-1, 3, C2f, [1024]]  # 21 (P5/32-large)

  - [[15, 18, 21], 1, Detect, [nc]]  # Detect(P3, P4, P5)
```

APPENDIX C
Training Code

importing libraries

```
from ultralytics import YOLO
from roboflow import Roboflow
import time
```

#downloading dataset

```
from roboflow import Roboflow
rf = Roboflow(api_key="**************")
project = rf.workspace("dhufr-farooq").project("physiogonomy")
dataset = project.version(1).download("yolov8")
```

#downloading the pretrained model (five model available “yolov8n.pt”, “yolov8s.pt”, “yolov8m.pt”, “yolov8l.pt”, “yolov8x.pt”)

```
model = YOLO("yolov8n.yaml").load("yolov8n.pt")
```

#model training

```
start_time = time.time()
model.train(data="path/to/custom_data.yaml", epochs = 200,
patience = 25,imgsz = 640,batch = 16 , pretrained = False)
end_time = time.time()
training_time = end_time-start_time
print("time taken is:{:.0f}m
{:.0f}s".format(training_time//60,training_time % 60))
```

#model validation (split=’test’ to val the model on test set)

```
 !yolo detect val model=path/to/model.pt  split='val'
```

#predict

```
infer = YOLO("path/to/model.pt")
infer.predict(source=image_path)
```

#exporting the model

```
cd path/to/exporting_folder
!yolo export model=path/to/model.pt
```

APPENDIX D
GUI Code

```
import ultralytics
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QLabel, QPushButton, QFileDialog
from PyQt5.QtGui import QPixmap
from ultralytics import YOLO
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("PHYSIOGNOMY")
        self.setGeometry(100, 100, 400, 400)
        # Create QLabel widgets to display the image and prediction
        self.image_label = QLabel()
        self.path_label = QLabel()
        # Create QPushButton widgets for image upload and display
        self.upload_button = QPushButton("Upload Image")
        self.upload_button.clicked.connect(self.upload_image)
        self.display_button = QPushButton("predict")
        self.display_button.clicked.connect(self.predict)
        # Create a central widget and layout to arrange the labels and buttons
        central_widget = QWidget(self)
        layout = QVBoxLayout(central_widget)
        layout.addWidget(self.upload_button)
        layout.addWidget(self.display_button)
        layout.addWidget(self.image_label)
        layout.addWidget(self.path_label)
        self.setCentralWidget(central_widget)
        self.image_path = None
    def upload_image(self):
```

```
    # Open a file dialog to select an image
    file_dialog = QFileDialog()
    self.image_path, _ = file_dialog.getOpenFileName(self, "Select Image", "", "Image Files (*.png *.jpg *.jpeg)")
    x = self.image_path
    if self.image_path:
        # Load the selected image and display it on the window
        pixmap = QPixmap(self.image_path)
        self.image_label.setPixmap(pixmap.scaled(300, 300))
def predict(self):
    infer = YOLO("C:/Users/Lenovo/Downloads/best.pt")
    image_path1=self.image_path
    res = infer.predict(source=image_path1,conf= 0.50)
    clist= res[0].boxes.cls
    cls = set()
    print(cls)
    for cno in clist:
        cls.add(infer.names[int(cno)])
        print(cls)
        text = list(cls)[:]
        if "oval_face" in text:
            class_1 = "oval face :rational, emotional, highly adaptable,long-sighted\n"
        else:
            class_1= ""
        if "rounded_nose" in text:
            class_2 = "rounded nose: generous, emotional, kindhearted,sensitive\n"
        else:
            class_2= ""
        if "rounded_face" in text:
            class_3 = "rounded face: compassionate, helpful, generous, considerate, honest,tolerant\n"            else:
            class_3= ""
        if "square_face" in text:
            class_4 = "square face: subjective, emulative, frank, honest, serious, responsible\n"
        else:
```

```
            class_4= ""
        if "straight_eyebrows" in text:
            class_5 = "straight eyebrows: stubborn, masculine, resourceful, decisive, faithful,upright in their action\n"
        else:
            class_5= ""
        if "rounded_eyebrows" in text:
            class_6 = "rounded eyebrows: gentle, wise, kind-hearted, mild-tempered, considerate,sensitive\n"
        else:
            class_6= ""
        if "blu_eye" in text:
            class_7 = "blue eye: youthfulness, peace, kind, spiritual people,good observation power\n"
        else:
            class_7= ""
        if "brown_eye" in text:
            class_8 = " brown eye: polite, affectionate self-confident, determined, persevering, spiritual,strong-headed\n"
        else:
            class_8= ""
        if "black_eye" in text:
            class_9 = "black eye: trustworthy, responsible, passionate, lively\n"
        else:
            class_9= ""
        if "pointy_nose" in text:
            class_10 = "pointy nose: clear thinking, tolerance, trustworthiness,tenacity\n"
        else:
            class_10= ""
        self.path_label.setText(class_1 +class_2 +class_3+class_4 +class_5 +class_6 +class_7 +class_8 +class_9 +class_10  )
if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

APPENDIX F
Modified Model Configuration File

```
# Parameters

nc: 6  # number of classes

scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml' will call yolov8.yaml with scale 'n'

  # [depth, width, max_channels]

[0.1, 0.1, 1024]  # summary : 211 layers, 2055490 parameters, and 2055474 gradients

# YOLOv8.0n backbone

backbone:

  # [from, repeats, module, args]

  - [-1, 1, Conv, [64, 3, 2]]  # 0-P1/2

  - [-1, 1, Conv, [128, 3, 1]]  # 1-P2/4

  - [-1, 3, C2f, [128, True]]

  - [-1, 1, Conv, [256, 3, 1]]  # 3-P3/8

  - [-1, 6, C2f, [256, True]]

  - [-1, 1, Conv, [512, 3, 2]]  # 5-P4/16

  - [-1, 6, C2f, [512, True]]

  - [-1, 1, Conv, [1024, 3, 2]]  # 7-P5/32

  - [-1, 3, C2f, [1024, True]]

  - [-1, 1, SPPF, [1024, 5]]  # 9


# YOLOv8.0n head

head:

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
```

- [[-1, 6], 1, Concat, [1]] # cat backbone P4

- [-1, 3, C2f, [512]] # 12

- [-1, 1, nn.Upsample, [None, 2, 'nearest']]

- [[-1, 4], 1, Concat, [1]] # cat backbone P3

- [-1, 3, C2f, [256]] # 15 (P3/8-small)

- [-1, 1, Conv, [256, 3, 2]]

- [[-1, 12], 1, Concat, [1]] # cat head P4

- [-1, 3, C2f, [512]] # 18 (P4/16-medium)

- [-1, 1, Conv, [512, 3, 2]]

- [[-1, 9], 1, Concat, [1]] # cat head P5

- [-1, 3, C2f, [1024]] # 21 (P5/32-large)

- [[15, 18, 21], 1, Detect, [nc]] # Detect(P3, P4, P5)

CURRICULUM VITAE

Dhufr AL-OBAIDI

EDUCATION

Master degree in computer science engineering
Gaziantep University
Gaziantep, Türkiye

Bachelor degree in network engineering
AL-Iraqia University
Baghdad, Iraq