

**FACE DETECTION AND RECOGNITION
IN AN UNCONSTRAINED ENVIRONMENT**

ARIANA CHIN SUE REI

UNIVERSITI TUNKU ABDUL RAHMAN

**FACE DETECTION AND RECOGNITION
IN AN UNCONSTRAINED ENVIRONMENT**

ARIANA CHIN SUE REI


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Honours) Electrical and Electronic Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

September 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Ariana Chin Sue Rei

ID No. : 1900599

Date : 11/9/2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“FACE DETECTION AND RECOGNITION IN AN UNCONSTRAINED ENVIRONMENT”** was prepared by **ARIANA CHIN SUE REI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic at Universiti Tunku Abdul Rahman.

Approved by,

Signature :



Supervisor :

IR. DR. THAM MAU LUEN
ASSISTANT PROFESSOR
LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN

Date :

28 Sep 2022

Signature :



DR. CHANG YOONG CHOON
HEAD
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN

Co-Supervisor :

Date :

2 Oct 22

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2022, Ariana Chin Sue Rei. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Tham Mau Luen for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my co-supervisor, Dr. Chang Yoong Choon, my loving parents, siblings and friends who had helped and given me encouragement to complete this project.

ABSTRACT

Face detection and recognition in an unconstrained environment is a challenging subject despite a plethora of solutions and the earnest efforts of numerous academics. Face refers to the front part of the head in humans from the forehead to the chin which includes the mouth, nose, cheeks, and eyes. Face detection refers to the ‘ability’ to identify faces whereas face recognition refers to the automated ‘technique’ used to confirm or identify a person based on physiological traits. Unconstrained environment refers to the lack of control over external variables such as illumination, position, occlusion, and distance from the camera. Face detection and recognition is a non-intrusive, discreet method of authenticating personal identification within enforcement agencies and business settings. While each face detection and recognition procedures is effective for the specific variant under investigation, performance declines quickly when other variances are present. In this project, five face datasets; (i)Labeled Faces in the Wild (LFW), (ii)Adience, (iii)Unconstrained Facial Images (UFI), (iv)Open Images V6 and (v)Unconstrained Face Detection Dataset (UFDD) with training set of 1500 images and test set of 300 images taken in an unconstrained environment (includes rain, snow, haze, blur, illumination and lens impediments) are manually annotated and trained on face detection models YOLOv4 and YOLOv5. It was found that YOLOv5 perform similarly with YOLOv4 using the mAP metrics. However, the training time of 6000 iterations for YOLOv5 is significantly lesser than YOLOv4. Additionally, YOLOv5 produces a much smaller weight file of 14 MB compared to YOLOv4. Open Images dataset performed the best with YOLOv5 model (86.1% mAP) for 300 test images taken in an unconstrained environment. The larger dataset LFW with greater number of labelled individual (1573 individuals) was able to achieve satisfactory results when verified with known or unknown faces in the database using the Siamese Neural Networks. The Siamese Neural Network can be further improved by increasing the number of verification images for each individual to reduce the likelihood of the model predicting a false positive. Finally, a prototype is developed using the face detection model, YOLOv5, using the weights obtained from Open Images dataset, and the face recognition Siamese Neural Network model, with the weights from LFW dataset.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	xii
LIST OF SYMBOLS / ABBREVIATIONS	xiii
LIST OF APPENDICES	xv

CHAPTER 1 1

1	INTRODUCTION	1
	1.1 Face detection	1
	1.2 Face recognition	2
	1.3 Unconstrained Environment	3
	1.4 Importance of Study	5
	1.5 Problem Statement	6
	1.6 Aim and Objectives	8
	1.7 Research Approach	9
	1.8 Project Approach	14
	1.9 Limitation of the Study	15
	1.10 Contribution of the Study	16
	1.11 Outline of the Report	16

CHAPTER 2 17

2	LITERATURE REVIEW	17
	2.1 Overview of Common Deep Learning Architectures	17
	2.2 Overview of Proposed Face Detection Models	21
	2.3 Overview of Face Recognition Algorithms	24
	2.4 Overview of Face Recognition Approach	28

2.5	Overview of Proposed Face Recognition Model	30
2.6	Overview of Face Databases	33
2.7	Overview of Platforms With Free Gpu And Cpu	35
CHAPTER 3	38	
3	METHODOLOGY AND WORK PLAN	38
3.1	Data Preparation and Annotation for Face Detection	38
3.2	Data Training and Testing Protocol	40
3.3	Face Detection using YOLOv4 and YOLOv5	41
3.4	Siamese Neural Network	44
3.5	Integration of Face Detection and Recognition	45
3.6	Gantt Charts	46
CHAPTER 4	47	
4	RESULTS AND DISCUSSION	47
4.1	Evaluation Techniques	47
4.2	Determine End of Training	49
4.3	YOLOv4 Results	50
4.4	YOLOv5 Results	52
4.5	Additional YOLOv4 and YOLOv5 Results	53
4.6	Comparison of YOLOv4 and YOLOv5 Results	54
4.7	Face Detection Result	55
4.8	Result for Face Recognition using Siamese Neural Network	57
CHAPTER 5	67	
5	CONCLUSIONS AND RECOMMENDATIONS	67
5.1	Conclusions	67
5.2	Recommendations for future work	68
REFERENCES		69
APPENDICES		73

LIST OF TABLES

Table 1.7.4:	Search results found for each website	10
Table 1.7.5:	Criteria to filter research materials	11
Table 2.7.1(a):	CPU-only VMs	35
Table 2.7.1(b):	GPU VMs	36
Table 2.7.1(c):	Execution and Idle Time	36
Table 4.1.1:	Definition of Terms Used in Face Detection Metrics	47
Table 4.3.1:	Metrics Result for YOLOv4 Data Training	50
Table 4.3.2:	Metrics Result for YOLOv4 Data Training (cont.)	50
Table 4.3.3:	Highest mAP achieved using YOLOv4	51
Table 4.4.1:	Metrics Result for YOLOv5 Data Training	52
Table 4.4.2:	Metrics Result for YOLOv5 Data Training (cont.)	52
Table 4.4.3:	Highest mAP achieved using YOLOv5	53
Table 4.5.1:	Train, test time and weights size for YOLOv4 and YOLOv5	53
Table 4.5.2:	Train, test time and weights size for YOLOv4 and YOLOv5 (cont.)	54
Table 4.7.1:	YOLOv5 Face Detection Result for Open Images Database and Adience	55
Table 4.7.2:	YOLOv5 Face Detection Result for UFDD, LFW and UFI	56
Table 4.8.1:	10 verification images of 10 individuals for comparison	59
Table 4.8.2:	Results obtained from known and unknown faces in the dataset using Siamese Neural Network	62
Table B-1:	Output of YOLOv4 data training for Open Images Dataset V6	74
Table B-2:	Output of YOLOv4 data training for Adience	76
Table B-3:	Output of YOLOv4 data training for Labeled Faces in the Wild (LFW)	78
Table B-4:	Output of YOLOv4 data training for Unconstrained Facial Images (UFI)	80

Table B-5:	Output of YOLOv4 data training for Unconstrained Face Detection Dataset (UFDD)	82
Table B-6:	Output of YOLOv5 data training for Open Images Dataset V6	84
Table B-7:	Output of YOLOv5 data training for Adience	84
Table B-8:	Output of YOLOv5 data training for Labeled Faces in the Wild (LFW)	85
Table B-9:	Output of YOLOv5 data training for Unconstrained Facial Images (UFI)	85
Table B-10:	Output of YOLOv5 data training for Unconstrained Face Detection Dataset (UFDD)	86

LIST OF FIGURES

Figure 1.1:	Standard factors to consider in face detection	1
Figure 1.3:	Sample images from UFDD dataset in seven different conditions	3
Figure 1.7:	Systematic review methodology	9
Figure 1.8:	Block diagram of face recognition system	14
Figure 2.1.1:	CNNs Architecture	17
Figure 2.1.2:	R-CNN Architecture	18
Figure 2.1.3:	SSD Architecture	19
Figure 2.1.4:	FPN Architecture	19
Figure 2.1.5:	GANs Architecture	20
Figure 2.2.1:	Architecture of YOLOv4 model	22
Figure 2.2.2:	Architecture of YOLOv5 model	23
Figure 2.4.1:	Simple illustration of a Siamese Neural Network	30
Figure 2.4.2:	Architecture of the Siamese neural network for face recognition	32
Figure 3.1(a):	Constructing bounding boxes in ‘makesense.ai’	38
Figure 3.1(b):	Annotation formats available in ‘makesense.ai’	39
Figure 3.1(c):	AI models available in ‘makesense.ai’	39
Figure 3.3.1	Block Diagram of Data Training using YOLOv4 with Darknet in Google Colab	41
Figure 3.3.2:	Face Detection Procedure using YOLOv4	42
Figure 3.3.3:	Block Diagram of Data Training using YOLOv5 with Pytorch on local device	43
Figure 3.4.1:	Siamese Neural Network Data Training	44
Figure 3.4.2:	Face Recognition Procedure using Siamese Neural Network	45
Figure 3.5:	Integration of Face Detection and Recognition Model	45
Figure 4.2:	Graph showing relationship between error and number of iterations	49
Figure 4.8.1:	Graph of total loss vs no. of epoch for 300 epochs for LFW(40)	57

Figure 4.8.2:	Graph of total loss vs no. of epoch for 100 epochs for LFW(1573)	58
Figure 4.8.3:	Face similarity on known face (Bill_Clinton)	64
Figure 4.8.4:	Face similarity on known face (Britney_Spears)	64
Figure 4.8.5:	Face similarity on unknown face (Liu_Yifei)	65
Figure 4.8.6:	Face similarity on unknown face (Kanye_West)	65
Graph A-1:	Mean Average Precision (mAP) vs Number of Iterations for YOLOv4	73
Graph A-2:	Mean Average Precision (mAP) vs Number of Iterations for YOLOv5	73

LIST OF SYMBOLS / ABBREVIATIONS

CNN	Convolutional Neural Network
R-CNN	Region Based Convolutional Neural Networks
SSD	Single Shot Detector
DNN	Deep Neural Network
FPN	Feature Pyramid Network
GAN	Generative Adversarial Network
HOG	Histogram of Oriented Gradients
LFW	Labeled Faces in the Wild
UFI	Unconstrained Facial Images
UFDD	Unconstrained Face Detection Dataset
HMM	Hidden Markov model
GPU	Graphics processing unit
CPU	Central processing unit
RAM	Random-access memory
GB	Gigabyte
VM	Virtual Machine
GHz	Gigahertz
SPP	Spatial Pyramid Pooling
PAN	Path Aggregation Network
BoF	Bag of Freebies
BoS	Bag of Specials
CIoU	Complete Intersection over Union
YOLOv4	You Only Look Once v4
YOLOv5	You Only Look Once v5
CSP	Cross Stage Partial Networks
d	Euclidean distance
$d(A, B)$	Euclidean distance between two input images A and B
$f(x^{(i)})$	Feature vector
$L(A, B, Y)$	Contrastive Loss Function
GT	Ground Truth
TP	True Positive

TN	True Negative
FP	False Positive
FN	False Negative
AP	Average Precision
P	Precision
R	Recall
mAP	Mean Average Precision

LIST OF APPENDICES

Appendix A: Graphs	73
Appendix B: Tables	74

CHAPTER 1

INTRODUCTION

1.1 Face detection

The term "face" describes the area of the human head that extends from the forehead to the chin and contains the mouth, nose, cheeks, and eyes. The "ability" to recognise faces is referred to as face detection. According to a 2021 survey by (Minaee, et al., 2021), face detection is a critical first step in tasks including face recognition, facial attribute categorization, face editing, and face tracking. Despite significant advancements in uncontrolled face identification in recent decades, reliable and efficient face detection in an unconstrained environment remains a challenge. Changes in positions, scale, lighting, facial expressions, picture distortion, face occlusion are some factors to be considered (Figure 1.1). Face detection, unlike conventional object detection, has fewer variances in aspect ratio but considerably higher variations in size.



Figure 1.1: Standard factors to consider in face detection (Yang, et al., 2016)

(Minaee, et al., 2021) further states that face identification attempts in the past were mostly focused on the traditional technique, where manually annotated features were taken from an image and passed into a classifier as input to find plausible face areas. The Haar Cascades classifier and the Histogram of Oriented Gradients (HOG), followed by SVM, are some classic works for face detection at that time.

In the last 6-7 years, researchers have developed various intriguing model architectures as a result of deep learning's remarkable success in computer vision. Most of the early deep-learning-based models used Cascade-

CNN architectures, which were inspired by the notion of a cascade of classifiers. However, recent deep-learning-based models favour single-shot detection, R-CNN based architectures, feature pyramid network (FPN) models, and others, thanks to the emergence of various unique architectures for generic object identification. (Minaee, et al., 2021)

Some prominent Deep Neural Network (DNN) architectures used are:

1. Convolutional Neural Networks (CNNs)
2. R-CNN Based Models
3. Single Shot MultiBox Detector
4. Feature Pyramid Network (FPN)
5. Generative Adversarial Networks (GANs)

1.2 Face recognition

Face recognition is an automated technique used to confirm or identify a person based on physiological traits. A biometric identification system, in general, uses physiological traits such as fingerprint and face or behavioural patterns such as handwriting, voice, or finger keystrokes to detect a person. Face recognition is a non-intrusive, discreet method of authenticating personal identification in a natural and civil manner. (Tolba, et al., 2006)

Two primary components of face recognition techniques are:

- (1) face detection, in sometimes congested environments, and normalisation to account for geometrical and lighting variations, and
- (2) face identification, based on the position and location of facial points.

The faces are identified using suitable classification algorithms, and the outputs are further processed through the utilisation of model-based systems and logistic feedback. Fully automated algorithms are those that have both components, while partially automatic algorithms have only the second portion. Face image and the location of the centre of the eyes are usually provided to partially automated algorithms whereas only facial images are required for fully automated algorithms. (Tolba, et al., 2006)

There are three categories of face recognition algorithms. They are profile, frontal, and view tolerant recognition. Although frontal recognition is

the most common method, view-tolerant algorithms normally employ a more sophisticated technique. Profile schemes, which are stand-alone systems, are only marginally useful for identification. They are, nevertheless, highly useful for quick pre-searches of extensive face databases in that is able to minimise the computational burden of a complex algorithm and a hybrid recognition system. To circumvent the shortcomings of the separate components, hybrid techniques integrate several recognition approaches in either a serial or parallel sequence. (Tolba, et al., 2006)

Facial recognition algorithms may be categorised based on models or exemplars. When dealing with appearance variation, models capture class information and give significant restrictions. Exemplars, on the other hand, can be employed for recognition. According to (Tolba, et al., 2006), present models-based procedures rarely utilise exemplar and contrariwise. This is due to the fact that these two techniques are not mutually exclusive.

1.3 Unconstrained Environment

Refers to the lack of control over external variables such as illumination, position, occlusion, and distance from the camera. Figure 1.3 displays a selection of photos from the UFDD dataset with typical categories of unconstrained environment such as rain, snow, haze, blur, illumination, lens impediments and distractors and its corresponding annotations.

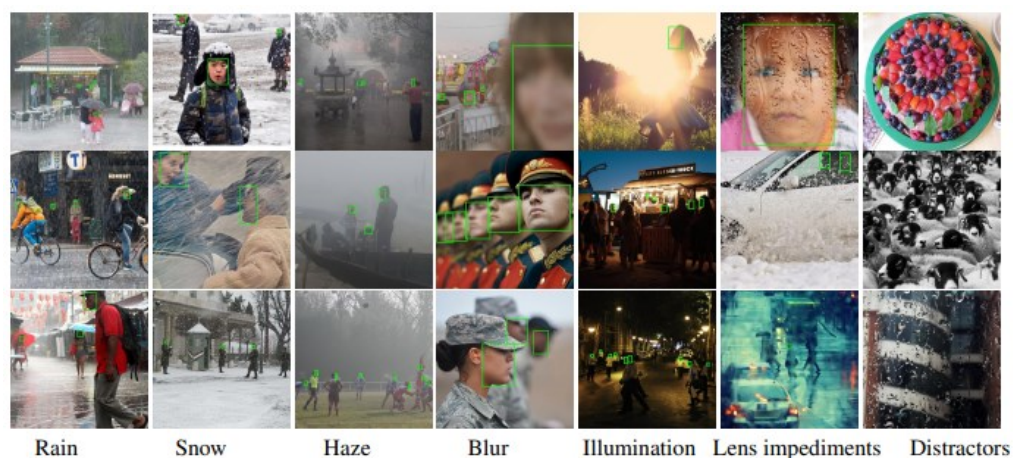


Figure 1.3: Example images originated from UFDD dataset
in seven different conditions (Nada, et al., 2018)

1.3.1 Rain

Water streaks disrupt the high-level frequency components of a picture, causing the filter responses to change. As a result, visual quality suffers and detection performance declines. Moreover, when photographs with occluded faces are further degraded with rain streaks, the situation becomes much more severe, making face detection difficult. (Nada, et al., 2018)

1.3.2 Snow

Snow, like rain, affects facial detection effectiveness by blocking particular portions of the face (as shown in Figure 1.3). Snow, on the other hand, causes a greater degree of opacity than rain, which might explain why the deterioration induced by snow is so much worse. (Nada, et al., 2018)

1.3.3 Haze

Low picture contrast, induced by the absorbance or scattering of light by drifting pollutants in the air, reduces the visibility of faces in photos. Haze not only degrades image quality but it also reduces face detection accuracy. Due to the existence of haze, the faces have reduced visual quality and tend to appear darker, as indicated in Figure 1.3. When compared to rain and snow, haze causes a greater reduction in performance. (Nada, et al., 2018)

1.3.4 Blur

Blurring of images may be caused by camera shaking or depth, leading to a loss of critical high-frequency components in a photo. Face identification is greatly hampered as a result of this loss of information. Existing face detectors' representations aren't resilient to foggy photographs because they are generally developed using datasets with clear, good image quality. (Nada, et al., 2018)

1.3.5 Illumination

Face visibility is affected by intense illumination situations such as extreme light or obscurity. In the study performed by (Nada, et al., 2018), all four methods (HR-ER, S3FD, SSH and Faster-RCNN) of face detection failed to recognise faces in photos with harsh lighting conditions.

1.3.6 Lens impediments

Debris or condensation of water on the camera lens causes the lens to be obstructed. It will induce abrupt discontinuities in frequency and, hence, causes substantial fluctuations in focus in the captured photos. The presence of water droplets causes areas of the picture to be out of focus. As a result, either erroneous or missed detections occur. (Nada, et al., 2018)

1.4 Importance of Study

Face recognition technology may be used in both enforcement agencies and business settings. In security applications such as criminal albums (stationary comparison) and security cameras (performing real-time identification), face recognition software is used. Commercial uses face recognition by performing stationary match of pictures on devices such as ATM cards, credit cards, and driver's licences for access control. (Tolba, et al., 2006)

Face recognition attendance systems are not reliant on a few facial traits, but rather on a number of data points to identify a person. As a result, these systems can detect face masks and identify persons without removing the mask or changing facial characteristics such as beards or specs. Employees do not have to remove their masks, which is a significant benefit over any other biometric method. Additionally, modern attendance systems include very precise face recognition algorithms that can detect changes in facial features such as spectacles, beards, and hats, among other things. (Truein, 2021)

Facial recognition can also be used to capture attendance in class for middle school, high school, college and universities. It ensures that students are present in class since they are required to scan their face to take attendance. Since each individual have unique facial features, there is no way students can seek help from other fellow students to take attendance for them. Hence, it can reduce the rate of truancy in schools. (Hoo & Ibrahim, 2019)

Similarly, facial recognition attendance system may be applied in industrial workplace. While the great majority of employees are trustworthy, buddy punching cannot be ruled out. Some people skip work and yet get paid by collaborating with coworkers or security officers. Such time fraud is not only harmful to businesses, but it is also unjust to hardworking employees. The facial recognition attendance system is not only able to take attendance, but it

can automatically record the employees' arrival and leave times. Hence, it also improves workplace security since the system can correctly identify who left the designated area and when. (Truein, 2021)

Pandemics like as Covid 19 can be better handled by limiting physical contact in public spaces and at work. There has been a substantial surge in demand and implementation of contactless technology since the epidemic. (Truein, 2021) For example, shops and outlets have stopped displaying 'tester' products due to hygienic safety. This indirectly discourages people from buying items such as makeup, skin care and others since it requires contact when trying free samples. Consequently, shoppers may find it difficult for them to identify whether the product is suitable for them or not through sight alone. Facial recognition technology can provide a platform to simulate how the products would look like by meshing the person's face with the product. For example, a shopper can simulate how a particular lipstick colour would look like on their face without directly applying the lipstick on their lips. (Balcazar, 2020)

Facial recognition only needs fewer contacts than other forms of security procedures, such as fingerprints. There is no need for physical interaction or direct social engagement. Rather, it is an automatic and seamless process through AI. Tasks such as entering locks and cellphones, withdrawing money from the bank or any other action that needs a combination lock, passcode or key can be made easier and safer through facial recognition technology. (Gargaro, 2021)

Early diagnosis of genetic abnormalities is another use of facial recognition technology. Facial recognition software may, in certain situations, detect how a particular syndrome is caused by specific genetic mutations just by analysing subtle facial traits. Using facial recognition technology could be much cheaper and faster than traditional genetic testing. (Gargaro, 2021)

1.5 Problem Statement

In general, face recognition is a challenging, unresolved subject despite a plethora of solutions and the earnest efforts of numerous academics. While each face recognition procedures is effective for the specific variant under investigation, performance declines quickly when further variances,

such as pose, scale, illumination changes, occlusion, etc., are present. A characteristic that is invariant to illumination, for example, may only work with consistent poses or facial expression, but drops in performance for changing pose or expression. For certain applications, such as management of entry to a secure area, this is not an issue because the training and test photos may be taken under comparable conditions. None of these approaches, however, are sufficiently robust for universal, unconstrained recognition. (Tolba, et al., 2006)

Even the finest systems today are confounded by changes in incident light, head attitude, face expression, haircut (including facial hair), cosmetics (including eyeglasses), and ageing. Despite the fact that several techniques for solving such challenges have been developed and shown to be effective, the obstacles persist. Because of these factors, automatic face recognition's matching performance is inferior to that of fingerprint and iris matching, despite the fact that it might be the most accessible measurement instrument for a certain activity. The failure margin of facial recognition are normally 2 to 25%. (Tolba, et al., 2006)

Furthermore, it is unclear whether various strategies can be integrated to overcome the shortcomings of one another. Depending on their structure, some strategies cannot be integrated with other algorithms. The Symmetric Shape-from-Shading approach of (Zheng, 2000), for example, is based on a frontal face's approximate symmetry. It is not clear how this approach may be paired with a side-profile based method, that is bereft of symmetry.

1.6 Aim and Objectives

This study aims to:

- develop a face detection and recognition system for static images and real time operation in an unconstrained environment, such as rain, snow, haze, blur, illumination and lens impediments.
- create custom annotations, perform face detection and evaluate the five publicly available datasets, (i)Labeled Faces in the Wild (LFW), (ii)Adience, (iii)Unconstrained Facial Images (UFI), (iv)Unconstrained Face Detection Dataset (UFDD) and (v)Open Images V6 on CNN-based object detection models YOLOv4 and YOLOv5.
- apply Siamese Neural Networks for single-shot face recognition using the datasets A&T and Labeled Faces in the Wild (LFW).
- integrate the face detection system with the face recognition system.
- develop a modular automatic face detection and recognition prototype based on the best performing combination.

1.7 Research Approach

With reference to (Shepley, 2019), this systematic review was performed using the following methodology, which is presented in Figure 1.7.

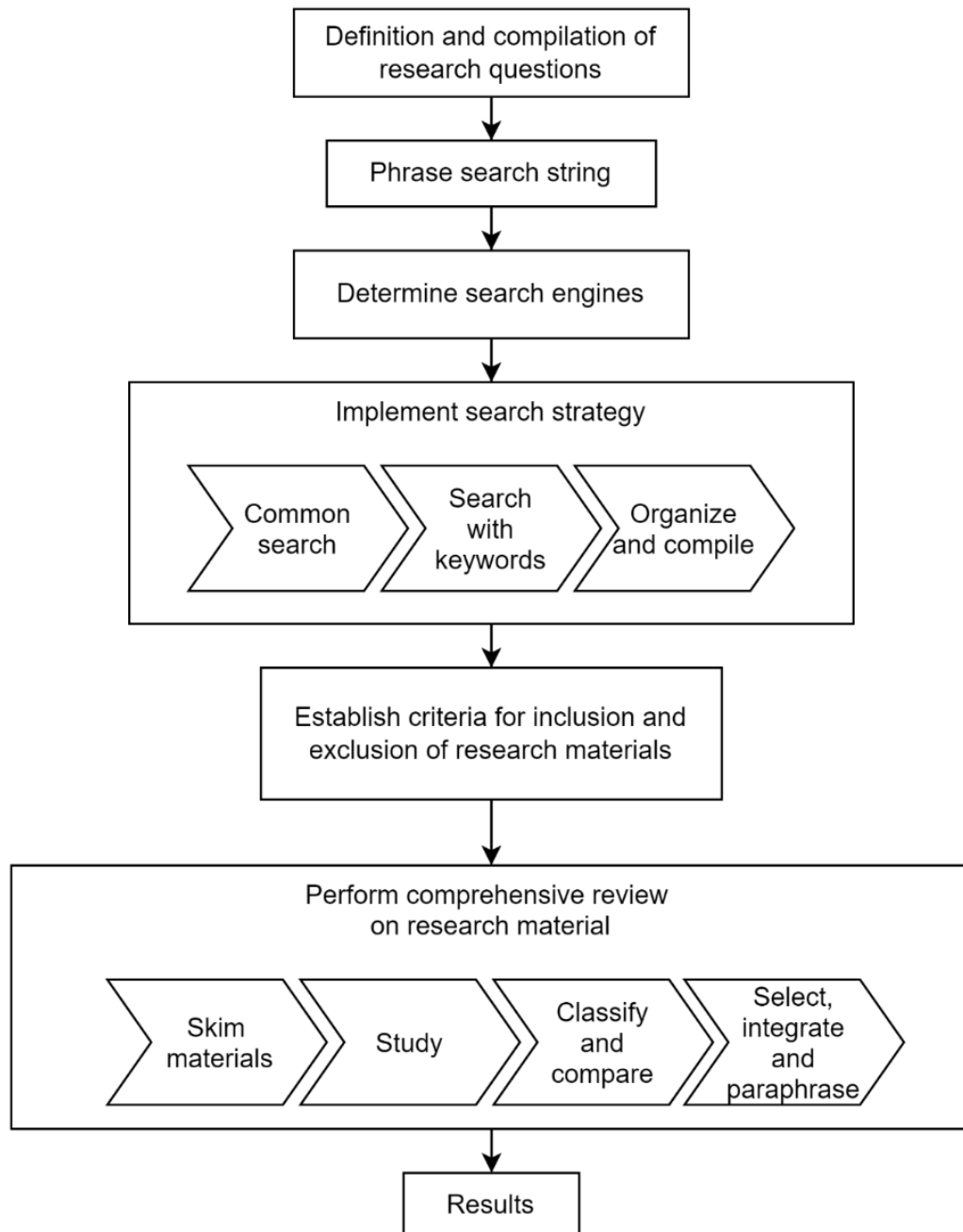


Figure 1.7: Systematic review methodology

1.7.1 Definition and compilation of research questions

Based on prior review and comprehension of facial recognition's current uses and applications, a set of study questions was formulated. These questions will attempt to develop a holistic picture of current developments in the field, study, identify, and assess state-of-the-art unconstrained facial recognition techniques, and bring to light the difficulties that current researchers face:

- (1) What are the most prevalent strategies for achieving unconstrained facial recognition, and which ones deliver good results?
- (2) What are the present issues of unconstrained facial recognition, and how can it be employed in the future?
- (3) Which datasets are being utilised to create and test unconstrained face recognition algorithms?
- (4) What are the most widely utilised unconstrained facial recognition tools?

1.7.2 Phrasing of search string

To ensure that the review yielded findings that clearly reflect trends and issues in current research, this study utilised these terms in both the common search and keywords search stages, only targeting publications published between 2000 and 2022. Initially, in the common search stage, individual phrases such as 'face recognition', 'face detection' and 'unconstrained environment' was used to study and understand the fundamentals of unconstrained face recognition.

In the second stage, combination of the following keywords, along with their synonyms, were discovered to be closely related to unconstrained face recognition:

- (1) Face OR Facial
- (2) Recognition OR Detection OR Identification OR Verification
- (3) Systems OR Techniques OR Algorithms OR Processes
- (4) Unconstrained OR Unrestricted OR 'in the wild'

1.7.3 Determine search engines

In this study, the research materials used mainly comprise of online news, blogs, thesis and journals. These research materials are available online and published in English. The following research materials were automatically and manually searched using:

- (1) Google (<https://www.google.com/>)
- (2) Google Scholar (<https://scholar.google.com/>)
- (3) IEEEXplore (<https://ieeexplore.ieee.org/>)
- (4) ScienceDirect (<https://www.sciencedirect.com/>)
- (5) SpringerLink (<https://link.springer.com/>)
- (6) Elsevier (<https://www.elsevier.com/en-xs>)
- (7) PapersWithCode (<https://paperswithcode.com/>)
- (8) Arxiv (<https://arxiv.org/>)
- (9) Github (<https://github.com/>)

1.7.4 Implementation of search strategy

All relevant journals, news article, conference proceedings, technical papers, and other relevant material were included in this search. All research materials were downloaded in pdf format for simple reference management and prevention of duplicated findings. The research was carried out between February 2022 to April 2022. Table 1.7.4 displays the results of the unrefined search.

Table 1.7.4: Search results found for each website

Website	Total number of search results
Google	1,410,000
Google Scholar	107,000
IEEEXplore	215
ScienceDirect	4,092
SpringerLink	8,404
Elsevier	51,396
PapersWithCode	8
Arxiv	36
Github	13

1.7.5 Establishment of criteria

This stage entailed determining which research will go to the next level of evaluation by using inclusion and exclusion criteria. The first step involved skimming and eliminating studies only on the basis of their names, where studies evidently did not fall within the project's scope. The title and abstract of each surviving study were utilised in the second step to assess conformity with the inclusion and exclusion criteria. The materials that passed all of the refining steps were then downloaded, and the introduction and conclusion were scrutinised using the same criteria. After passing this last stage, all materials were reviewed in their entirety and included or removed based on the same criteria.

Table 1.7.5: Criterias to filter research materials

Inclusion	Exclusion
Include only English materials.	Exclude non-English materials.
Include materials that are distinct from other materials.	Exclude duplicated materials.
Only studies that specifically cover unconstrained facial recognition are included.	Studies that are not related to face recognition or are solely about constrained face recognition are excluded.
All journals, conference papers and technical publications should be included.	Very brief articles, posters, presentations, and editorials should be excluded.
All material that answers one or more research questions should be included.	Materials that do not provide a response to at least one of the research questions should be excluded.
Prioritize journals, conference papers and technical publications with sample code in Github.	Discard journals, conference papers and technical publications without sample code in Github.
All research published between 2000 and 2022 are included.	All research published before 2000 are excluded.

1.7.6 Perform comprehensive review

Following the execution of the search string and the implementation of the inclusion and exclusion criteria, the search results yielded the set of refined main studies. The collection of studies is further narrowed down to about 25 papers and all of them are used for data extraction and assessment in accordance with the research questions. Finally, the necessary data is selected, integrated and paraphrased to help achieve the objectives of the project.

The evaluation is based on the following data types retrieved from each materials:

- (1) Title
- (2) Year
- (3) Facial detection algorithm
- (4) Facial identification algorithm
- (5) Face database used
- (6) Type of unconstrained environment addressed
- (7) Accuracy of facial recognition system

1.8 Project Approach

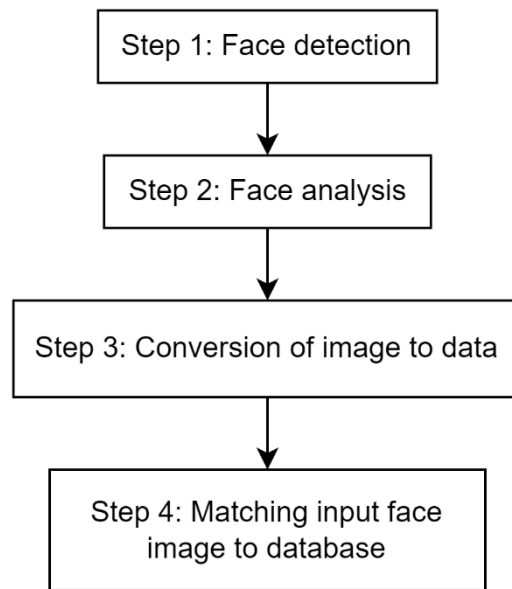


Figure 1.8: Block diagram of face recognition system

1.8.1 Step 1: Detecting Face

The system first identifies and detects the picture of a face whether it belongs to one person or multiple at once. The pose of individual in the photograph might be posing directly in front of the photographer or a profilic picture. (kaspersky, n.d.)

1.8.2 Step 2: Analysing Face

A facial picture is captured and examined. Most facial recognition technology uses 2D images because it is much easier to match them with photos in a database compared to 3D images. The programme analyses the geometry of the face. The gap between the eyes, the eye socket's depth, length between the forehead and chin, the curve of the cheekbones, lip's contour and others need to be considered. Ultimately, it is required to determine the facial landmarks that are essential for distinguishing the face. (kaspersky, n.d.)

1.8.3 Step 3: Conversion of image to data

After capturing the face image, the analogue data of the face is transformed into digital data. The facial examination is simplified to a technique that can be

calculated analytically. This code is known as a faceprint. Each person's faceprint is distinct in the same way as thumbprints are. (kaspersky, n.d.)

1.8.4 Step 4: Matching input face image to database

After that, the faceprint is compared to a database of previously recognised individuals. For example, any person tagged in a photo on Facebook is saved in Facebook's database and may be used for facial recognition. A determination is made if the faceprint matches exactly to an image in the facial recognition database. (kaspersky, n.d.)

1.9 Limitation of the Study

Most face recognition system can only cater specific variant under investigation. As a result, their performance declines quickly when further variances, such as pose, scale, illumination changes, occlusion, etc., are present. According to (Tolba, et al., 2006), none of these approaches are sufficiently reliable for universal, unconstrained recognition. It is difficult to find research materials and source code on Github that features face recognition in an unconstrained environment. Furthermore, it is uncertain if different approaches may be used to overcome one another's flaws. Some techniques cannot be used with other algorithms due to their structure. (Tolba, et al., 2006)

1.10 Contribution of the Study

In this paper, five publically accessible datasets—Labeled Faces in the Wild (LFW), Adience, Unconstrained Facial Images (UFI), Unconstrained Face Detection Dataset (UFDD), and Open Images V6—are analysed and evaluated using CNN-based object detection models YOLOv4 and YOLOv5. A workflow of the face detection process for both YOLOv4 and YOLOv5 is also constructed using custom data training. Additionally, the precision, recall and mean average precision of the datasets trained on YOLOv4 and YOLOv5 for train set of 1500 images and 300 test images for 6000 iterations are recorded and used to make comparison between the two models YOLOv4 and YOLOv5. This study also offers a workflow for the Siamese Neural Network facial recognition algorithm with customised data training. Moreover, this study successfully integrates the YOLOv5 face detection model with Siamese neural network face recognition model to create a face detection and recognition system that runs sequentially.

1.11 Outline of the Report

- Perform literature review on face detection and recognition in unconstrained environment.
- Perform literature review on face database used.
- Perform literature review on proposed face detection model YOLOv4 and YOLOv5.
- Perform literature review on proposed face recognition model using Siamese Neural Network.
- Plan and carry out workplan for face detection using YOLOv4 and YOLOv5.
- Plan and carry out workplan for face recognition using Siamese Neural Network.
- Perform analysis and comparison for face detection model.
- Selection of best performing face detection model with dataset.
- Integration of the face detection model with face recognition model to create a face detection and recognition system that runs sequentially.
- Recommend future improvements and plan.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of Common Deep Learning Architectures

This section covers a summary of deep learning architectures such as Convolutional Neural Networks (CNNs), Regional CNN (R-CNN), Single Shot Detector (SSD), Feature Pyramid Network (FPN) and Generative Adversarial Networks (GANs).

2.1.1 Convolutional Neural Networks (CNNs)

CNNs is a popular and common deep learning architectures, usually used for computer vision problems. (Minaee, et al., 2021) Typically, CNNs have three types of layers (Figure 2.1.1):

- i) convolutional layers to extract features through the convolution of a kernel of weights.
- ii) nonlinear layers which applies an activation function to feature maps and model nonlinear tasks.
- iii) lowering spatial resolution by pooling layers through the replacement of small neighbourhoods in a map with statistics of those neighbourhoods.

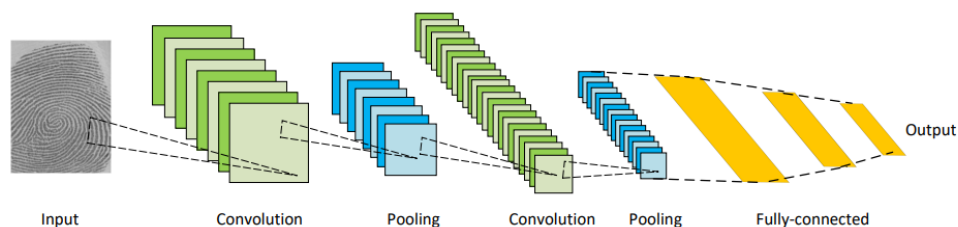


Figure 2.1.1: CNNs Architecture (Minaee, et al., 2021)

Layers of neural units are locally coupled. Each unit obtains weighted inputs from a small neighbourhood of units from the preceding layer. Layers at a higher level learn characteristics from growing receptive fields by assembling layers to produce multi-resolution pyramids. The fundamental computational benefit of CNNs is that the weights are distributed among

receptive fields in every layer, leading to much fewer parameters than fully convolutional neural networks. AlexNet, GGNet, and ResNet are some popular CNN architectures. (Minaee, et al., 2021)

2.1.2 Regional CNN (R-CNN)

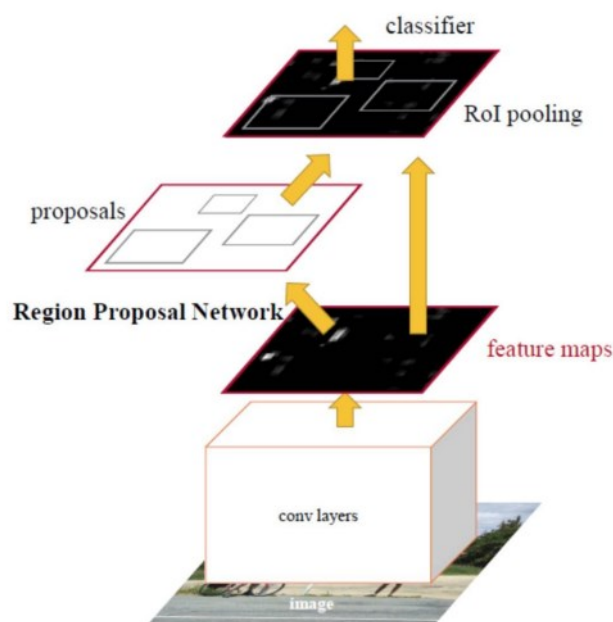


Figure 2.1.2: R-CNN Architecture (Minaee, et al., 2021)

A region proposal network (RPN) is used in the Faster R-CNN architecture (Figure 2.1.2) to create bounding box annotations. First, RPN creates a Region of Interest (RoI). Then, the proposed candidates are evaluated by a RoIPool layer to identify the object's class and the coordinates of the enclosed box. R-CNN extensions have also addressed the instance segmentation problem. Instance segmentation problem refers to the ability to execute object identification and segmentation at the same time. (Minaee, et al., 2021)

2.1.3 Single Shot Detector (SSD)

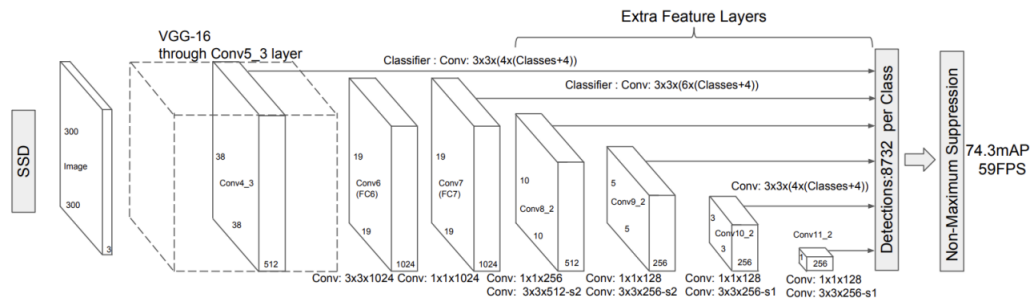


Figure 2.1.3: SSD Architecture (Minae, et al., 2021)

The resulting region of bounding boxes is separated into a collection of predefined boxes with varied aspect ratios and sizes. At prediction time, the network calculates ratings for the occurrence of each item category in each default box, and then modifies the box to match the object shape better. Moreover, the network uses predictions from a variety of feature maps with varying qualities to naturally handle objects in different sizes. SSD is simpler than approaches that based on classifier since it eliminates the formulation of proposals and following pixel or feature subsampling processes, consolidating all computation in a network segment. (Minae, et al., 2021)

2.1.4 Feature Pyramid Network (FPN)

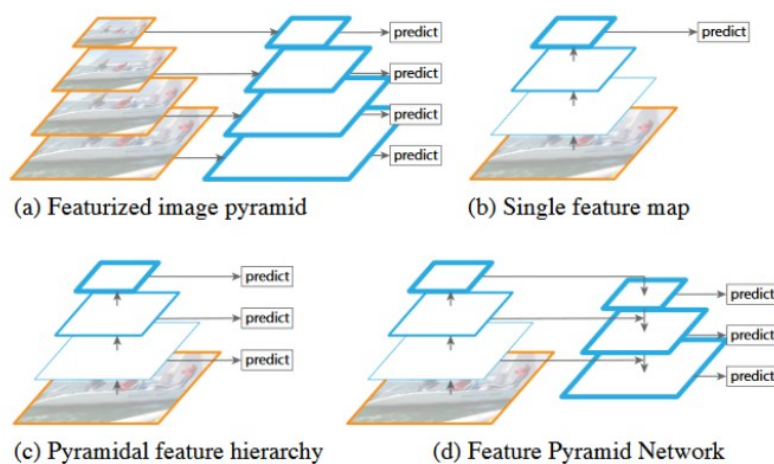


Figure 2.1.4: FPN Architecture (Lin, et al., 2017)

Object recognition algorithms that recognise items of varied sizes use feature pyramids as a key component. (Lin, et al., 2017) constructed feature pyramids at a minimal extra cost in their study using inherent multi-scale, pyramidal structure of deep convolutional network. A top-down architecture with lateral connections was developed for creating high-level semantic feature maps at all sizes (called FPN) and shown that FPN can enhance numerous vision tasks significantly.

2.1.5 Generative Adversarial Networks (GANs).

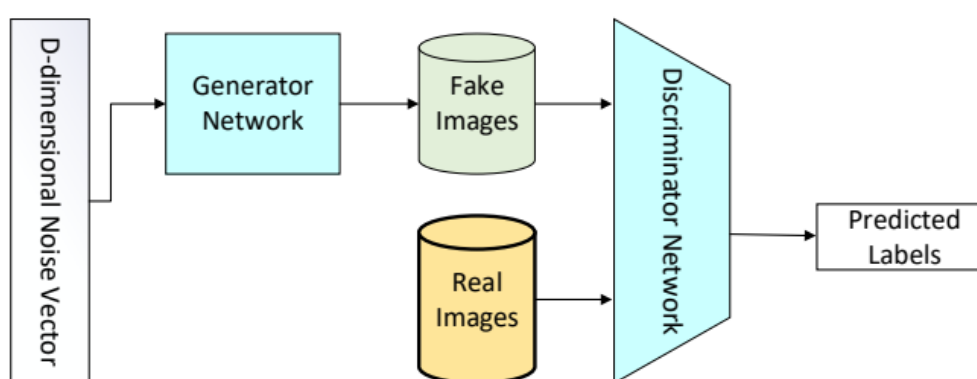


Figure 2.1.5: GANs Architecture (Minaee, et al., 2021)

Two networks; a generator and a discriminator (Figure 2.1.5) makes up GAN. In a typical GAN, the generator network G learns to map distortion z to a distributed receiver y that is comparable to "real" data. The discriminator network D distinguishes between manufactured "fake" inputs and real values. GAN training may be thought of as a competition between G and D , in which D tries to lower its overfitting while distinguishing fake from real data. As a result, the error rate is maximised. G , on the other hand, aims to reduce the loss function by increasing the discriminator network's inaccuracy. (Minaee, et al., 2021)

2.2 Overview of Proposed Face Detection Models

This section covers a summary of face detection models YOLOv4 and YOLOv5 which uses the deep learning architecture ‘Single Shot Detector (SSD)’.

2.2.1 YOLOv4 (You Only Look Once v4)

The main author of YOLOv4 is Alexey Bochkovskiy. YOLOv4 is a one-stage object detector that has been enhanced and improved over YOLO. The YOLOv4 model comprises of three components; backbone, neck, and head. (Figure 2.2.1) The feature extractor model, CSPDarknet53, is employed in the model's backbone. In the neck section of the model, Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PAN) are employed. In YOLOv4, modified PAN was employed for segmentation purposes. They used the concatenation operation instead of the addition procedure to modify PAN. The SPP is used to perform maximum pooling over a feature map. The accuracy of the model is improved by combining feature maps with the concatenation technique. The head component of YOLOv4 was kept the same as it was in YOLOv3. (Protik, et al., 2021)

In the YOLOv4 research, (Bochkovskiy, et al., 2020) concentrated on enhancing the existing version's accuracy and speed. They employed two strategies to increase accuracy and speed; Bag of Freebies (BoF) and Bag of Specials (BoS). BoF approaches assist to enhance accuracy without increasing the cost of inference. The creators of the BoF category have developed Mosaic, a new data augmentation approach. In the mosaic data augmentation approach, they blended four separate photos into one single image. The DropBlock regularisation technique is also utilised in YOLOv4.

DropBlock is an organised dropout method that improves the accuracy of YOLOv4. YOLOv4 also took advantage of the CIoU (Complete Intersection over Union) loss by utilising them for bounding box regression loss.

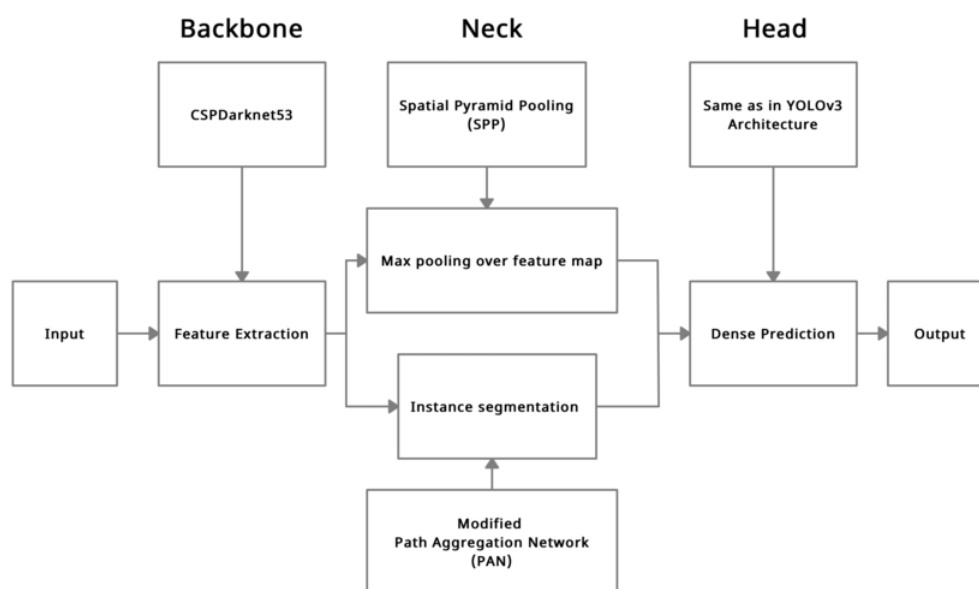


Figure 2.2.1: Architecture of YOLOv4 model (Protik, et al., 2021)

2.2.2 YOLOv5 (You Only Look Once v5)

Glenn Jocher is credited as the creator of YOLOv5, however all of the code is stored in the Ultralytics LLC repository. The majority of YOLOv5's performance gain comes from PyTorch training techniques, but the model architecture is still quite similar to YOLOv4. These improvements were initially known as YOLOv4, however as YOLOv4 was only recently released in the Darknet framework, the name was changed to YOLOv5 to prevent version conflicts. The repository contains the following 4 models: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. (Solawetz, 2020)

YOLOv5's most significant contribution is the translation of the Darknet research framework to the PyTorch framework. The Darknet framework, which is mostly written in C, provides precise control over the network operations. The ability to manage lower level language is beneficial to research in many ways, but it also slows down the incorporation of newer study findings. Additionally, the PyTorch framework supports reducing the 32 bit floating point precision in training and inference to 16 bit precision. As a result, the YOLOv5 models' inference time is greatly sped up. YOLOv5 creates model configuration in .yaml rather than Darknet's .cfg files. The key distinction between these two file types is that the .yaml file is compressed so

that it only contains information about the network's various levels, which are then multiplied by the number of layers in the block. (Solawetz, 2020)

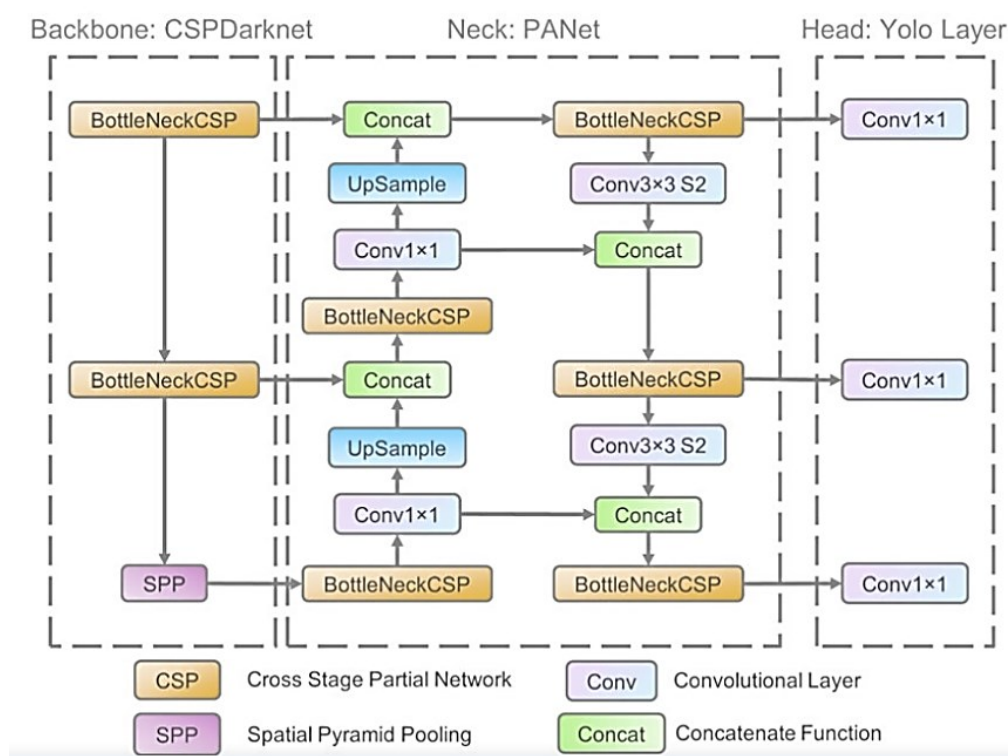


Figure 2.2.2: Architecture of YOLOv5 model (Garg, 2021)

Referring to Figure 2.2.2, the architecture of YOLOv5 consists of three components; backbone, neck and head. (Garg, 2021)

1. **Backbone:** Extract important characteristics from an input picture. In YOLO v5, CSP(Cross Stage Partial Networks) are utilised as the framework to extract highly beneficial features from an input picture.
2. **Neck:** Used to construct feature pyramids to assist models to effectively generalise. The use of feature pyramids helps models operate efficiently on previously unexplored data and makes it easier to recognise the same thing in a range of scales and sizes. In YOLO v5, PANet is utilised as a neck to get feature pyramids.
3. **Head:** Performs final detection. It employs anchor boxes to generate final output vectors with class probabilities, objectness scores, and bounding boxes.

2.3 Overview of Face Recognition Algorithms

This section covers a summary on the major human face recognition techniques that apply mostly to frontal faces such as eigenfaces, neural networks and hidden Markov models (HMMs).

2.3.1 Eigenfaces

One of the most thoroughly researched face recognition algorithm is eigenface. Mathematically, eigenfaces represents the primary components of the distribution of faces, or the eigenvectors of the covariance matrix of the collection of face pictures. The eigenvectors are arranged in order to indicate varying degrees of variance between the faces. A linear combination of the eigenfaces may be used to accurately represent each face. Only the "best" eigenvectors with the biggest eigenvalues can be used to estimate it. The finest M eigenfaces will create an M -dimensional space, sometimes known as "face space." (Nada, et al., 2018)

Since images usually contain a substantial amount of background region, face recognition systems must evaluate performance under varying lighting situations by establishing correlation between pictures with variations in illumination. However, a study by (Grudin, 1997) using eigenfaces, shows that the correlation between whole-face pictures is insufficient for satisfactory recognition performance. (Sirovich, 1990) also mentions that the eigenfaces technique also necessitates illumination normalisation.

Therefore, although eigenface algorithm appears to be a quick, easy, and practical approach, it does not produce invariance when the size and illumination circumstances vary.

2.3.2 Neural Networks

It is possible that the appeal of employing neural networks stems from the network's nonlinearity. A single layer adaptive network dubbed WISARD, which incorporates a different network for each stored individual, was one of the earliest artificial neural networks (ANN) approaches utilised for facial recognition. For effective recognition, the method for creating a neural network structure is critical. It depends extremely on the intended function. (Nada, et al., 2018)

(Lawrence, et al., 1997) developed a hybrid neural network combining local image sampling, a self-organizing map (SOM) neural network, and a convolutional neural network. The SOM quantizes picture samples into a topological space in which inputs that are close together in the original space are similarly close together in the output space, resulting in dimension reduction and invariance to slight changes in the image sample. Using an ORL database of 400 photos of 40 people, the authors reported 96.2 percent accurate recognition.

In another study, (Lin, et al., 1997) used a probabilistic decision-based neural network (PDBNN), modified from decision-based neural network (DBNN). The PDBNN may be used as

- 1) Face detector: locate a human face in a crowded picture.
- 2) Eye localizer: identify the locations of both eyes to create relevant feature vectors.
- 3) Face recognizer.

PDBNN is used to divide the network into K subnets. Each subset is used to locate a specific person in the dataset. The Gaussian activating method is implemented by PDNN for its neurons and the weighted aggregation of the neuron outputs is the output of each "facial subnet". To rephrase, the face subnet uses the mixture-of-Gaussian model to evaluate the probability density in the face space. (Lin, et al., 1997)

The PDNN's learning method is divided into two stages. Firstly, each subnet is trained using a distinct set of face pictures. The network variables are taught using particular samples from distinct face classes in the following stage, known as decision-based understanding. When a sample is misclassified and

allocated to the wrong subdomain, the right subdomain changes its settings to bring its selection closer to the miscategorized sample. According to (Lin, et al., 1997), the PDBNN face recognizer could recognise up to 200 people with 96% correct recognition rate in less than a second. As the number of people grows, however, the cost of computation also increases.

As a result, when the class quantity (i.e., people) increases, neural network approaches frequently encounter problems. Furthermore, because the systems must be trained to "optimal" parameter values using several model images per individual, they are unsuited for a single system image identification test. (Lin, et al., 1997)

2.3.3 Hidden Markov Models (HMMs)

For voice applications, stochastic modelling of nonstationary vector time series based on (HMM) has proved particularly effective. This approach was used to recognise human faces in a study by (Samaria, et al., 1994). Corresponding to hidden Markov model states, faces were intuitively split into areas such as the eyes, nose, mouth, and so on. Images should be transformed into 1D temporal or 1D spatial sequences because HMMs need a one-dimensional observation sequence and pictures are two-dimensional.

In a paper by (Samaria & Harter, 1994), using a band sampling approach, a spatial observation sequence was recovered from a face image. A 1D vector series of pixel observations was used to represent each facial image. Each observation vector is made up of a block of L lines, with a M line overlapping between them. An observation series is first sampled from an unknown test picture. After that, it is compared to every HMM in the model face database (different subject is represented by each HMM). The best match is the one with the highest probability, and the applicable model discloses the test face's identity.

Using the ORL (Our Database of Faces) database, which has 400 photos of 40 persons, the HMM technique has an recognition rate of 87%. In the preliminary test performed by (Samaria & Harter, 1994), a pseudo 2D HMM was found to obtain a 95% recognition rate. Its categorization and training periods though not specified was presumed to be costly. Additionally, the parameters were chosen based on subjective intuition. (Tolba, et al., 2006)

2.4 Overview of Face Recognition Approach

This section aims to concentrate and compare the two main face recognition approach called face verification and pair matching.

2.4.1 Face Verification

In the face verification paradigm, there is a pre-defined gallery of face photos of a group of people, each of whose identification is known. The challenge is to select a fresh query image and determine which individual in the gallery it represents. For example, suppose the gallery contains ten photographs of ten distinct individuals, and the job is to determine which of the ten persons each new input image represents. (Huang, et al., 2007)

For some applications, such as security access identification, where gallery photographs may be taken ahead of time in a fixed context and query images can be taken in the same environment, this assumption is feasible. This assumption, however, does not hold true for a wide range of jobs. For example, as part of an information retrieval assignment, a user may prefer to have images automatically tagged with the names of individuals, utilising a gallery of previously manually marked photographs that were not shot in a controlled setting. (Huang, et al., 2007)

2.4.2 Pair Matching

The pair matching paradigm is an alternative formulation of face recognition whereby for a given pair of face pictures, the face recognition system must determine whether they belong to the same person. There are a variety of subtle but significantly distinct recognition challenges within the pair matching paradigm. Some of the discrepancies have to do with how the database's training and testing subsets are structured. To put it another way, none of the persons depicted in any pair of photographs in the training set should appear in any of the test set pairs. In the same way, no test picture should exist in the associated training set. The unseen pair match problem refers to a situation in which neither of the persons depicted in the test pair has been observed during training. (Huang, et al., 2007)

It is basically impossible to develop a model for every individual in the test set during training, which distinguishes this challenge from the face verification paradigm. Because the persons in test photographs have never been seen before, there is no way to develop models for such individuals other than during testing from a single image. Rather, this paradigm is intended to focus on the general difficulty of distinguishing any two persons who have never been seen before. As a result, rather than learning to locate exemplars of a gallery of individuals as in face verification, a different sort of learning should be proposed that is learning to differentiate among any pair of faces. (Huang, et al., 2007)

2.4.3 Proposed Face Recognition Approach

Unseen pair matching, according to (Huang, et al., 2007), is one of the most common and fundamental face recognition challenges. Humans can recognise faces after just seeing one example image, which distinguishes them from computers that can only match against a predefined gallery of exemplars. Furthermore, as endeavours to scale recognition systems to handle orders of greater magnitude of people increases, algorithms built to learn generic variability will be less computationally and resource heavy than approaches that seek to learn a customised model for each person, and will almost certainly perform better as well.

Additionally, pair matching algorithms require less supervision since they simply require instances of matching and mismatching pairs to be found, rather than exemplars of each individual. Picture annotation problem would be greatly simplified as a result of this. By grouping face images that are likely to be the same person, a pair matching algorithm could be trained independently on different existing data and then used to identify photographs in a collection with the names of the persons depicted. A face verification algorithm, on the other hand, would need to be trained on manually labelled instances and would only be able to distinguish persons from the labelled examples. (Huang, et al., 2007) Hence, for this project, the pair matching approach is much more appropriate for the five different database in developing a face recognition

system in unconstrained environment. A Siamese neural network is proposed as the face recognition model.

2.5 Overview of Proposed Face Recognition Model

Unlike a typical CNN, the Siamese Network primarily calculates the distance between any two particular photos, rather than categorising or labelling the images. The network learns the parameters, such as weights and biases, and creates a lower distance if the two images belong to the same person. In contrast, the distance should be greater if they belong to different persons.

2.5.1 Siamese Neural Network

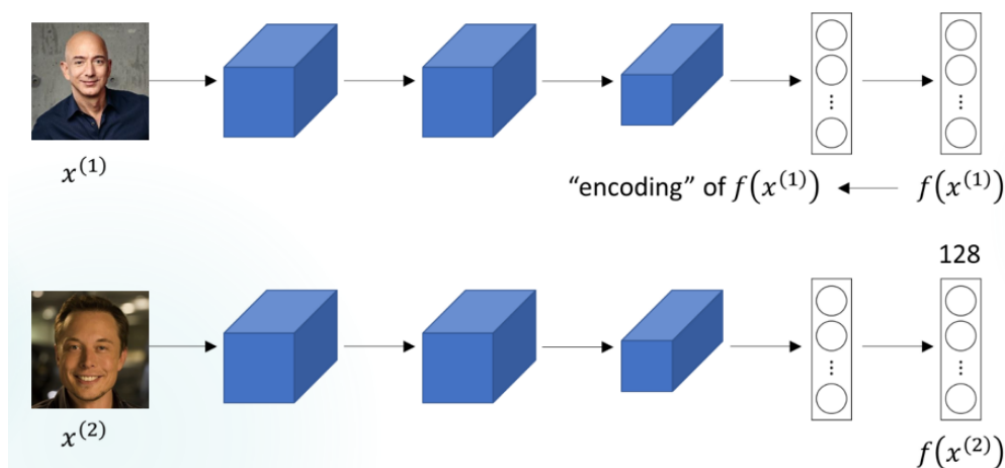


Figure 2.4.1: Simple illustration of a Siamese Neural Network
(Stefanovic, 2021)

Referring to Figure 2.4.1, the two input photos, $x^{(1)}$ and $x^{(2)}$, are processed to the conventional convolutional layers, maximum pooling, and fully connected layers in order to produce feature vectors. These feature vectors will subsequently be utilised as inputs to determine the degree of similarity between the two photos. These feature vectors are often passed into a softmax algorithm to do classifications. However, the Siamese neural network compares these two vectors instead of passing them through a softmax function. The two vectors should still be roughly the same if they belong to the same person but

have different head poses in the two photographs. These two vectors, on the other hand, should be different if they are of different persons. (Stefanovic, 2021)

One intriguing aspect of Siamese Neural Networks is that they produce two feature vectors simultaneously by utilising two identical neural networks. Using a new metric called d , L2 norm, which computes the euclidean distance between the two vectors, are passed through the metric to determine how dissimilar the two vectors are. Hence, to train the neural network, the vectors must be transformed so that the encoding it produces results in a function d that determines how similar the two pictures are. (Stefanovic, 2021)

A feature vector, or encoding $f(x^{(i)})$, is determined by a neural network's parameters. This indicates that the network will produce a feature vector with 128 elements when given any picture $x^{(i)}$. When two photos of the same person, $x^{(i)}$ and $x^{(j)}$, are used in a neural network, these parameters are trained such that the difference between the encoding should be minimal. In contrast, the difference will be large if the two images do not belong to the same person. (Stefanovic, 2021)

2.5.2 Contrastive Loss Function

The euclidean distance, d , between two input images A and B are given as

$$d(A, B) = ||f(A) - f(B)||^2$$

The equation for euclidean distance, d , is modified to accommodate a certain margin, m , to become

$$d(A, B) = \max(0, m^2 - ||f(A) - f(B)||^2)$$

The goal of the Siamese network is to move the feature vector representation of the input pictures with the same label closer and the input images with different labels further apart. Then, the two equations are combined together to form the Contrastive Loss Function

$$L(A, B, Y) = (Y) * ||f(A) - f(B)||^2 + (1 - Y) * \{\max(0, m^2 - ||f(A) - f(B)||^2)\}$$

where Y represents the label. A label of 1 indicates that the two input images belong to the same person and 0 indicates the images belong to different persons.

If the input images are from the same individual (positive pair), Y is 1. Only the first component of the Contrastive Loss Function will be considered since the second component will become zero due to the $(1-Y)$. This allows the distance between the two embeddings to be as minimal as possible if the two input images belong to the same individual. Similarly, for a negative pair, Y is 0 and only the second component of the Contrastive Loss Function is considered since the first component will be omitted. Consequently, the distance between the two embeddings will become larger and larger until it reaches the margin m . The result of the Contrastive Loss Function will be a value indicating whether or not the two photographs belong to the same individual. Figure 2.4.2 depicts the full architecture of the Siamese neural network that will be used in this project for face recognition.

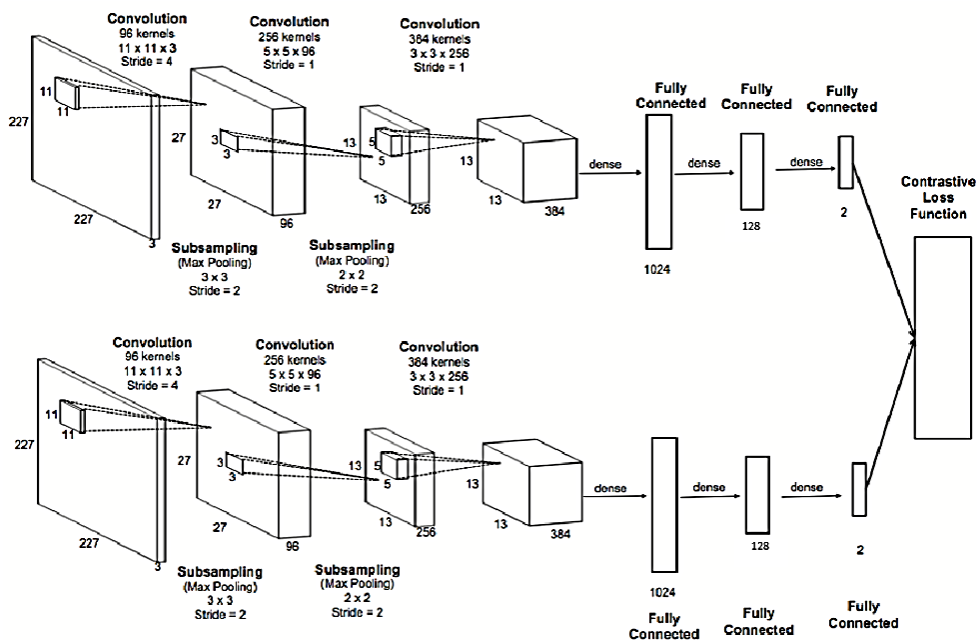


Figure 2.4.2: Architecture of the Siamese neural network for face recognition

2.6 Overview of Face Databases

This section covers a summary of face databases that includes images taken in unconstrained environments such as Labeled Faces in the Wild (LFW), Adience, Unconstrained Facial Images (UFI) and Unconstrained Face Detection Dataset (UFDD). Open Images Dataset V6 is used as a benchmark face database.

2.6.1 Labeled Faces in the Wild (LFW)

Labeled Faces in the Wild is a database of face photographs created to investigate the topic of unconstrained face recognition. Over 13,000 photos of faces were gathered from the internet for the database. The name of the individual pictured has been labeled on each face. In the database, 1680 of the persons shown have two or more photographs. Unfortunately, many ethnic groups are underrepresented in LFW. There are also few children, no infants, very few adults above the age of 80, and a small proportion of women. Furthermore, several ethnic groups have very little or no representation. Poor lighting, exaggerated poses, heavy occlusions, low resolution, and other crucial aspects are also not a big component of LFW. (Huang, et al., 2007)

2.6.2 Adience

This dataset contains 26,580 photos of 2,284 people, which are divided into eight age categories, genders, and subject labels (identity). The data in this collection is meant to be as accurate as possible to the constraints of real-world imaging. It tries to capture all of the variances in appearance, noise, stance, lighting, and other factors that might occur when photographs are taken without proper preparation or posing. The photographs in this set are from Flickr albums, which were produced via automated upload from iPhone5 or later smartphone devices and released to the public under the Creative Commons (CC) licence by their creators. It constitutes the biggest, completely unconstrained image collection for age, gender, and subject recognition. (The Open University of Israel and Adience, 2014)

2.6.3 Unconstrained Facial Images (UFI)

Unconstrained Facial Images (UFI) is a real-world database that includes images derived from genuine photographs taken by Czech News Agency (TK) reporters. It is primarily designed to benchmark facial recognition algorithms, but it may also be used for a variety of other applications (e.g. face detection, verification, etc.). The database is available in two separate parts. The first part consists of cropped faces that were automatically retrieved from the photographs using the Viola-Jones technique. As a result, the face sizes are nearly identical, and the photographs only have a minimal amount of backdrop. The photos in the second division contain a lot more backdrop, different face sizes and the faces are not localised. The name of a person is annotated on each image. (Lenc & Kral, 2015)

2.6.4 Unconstrained Face Detection Dataset (UFDD)

According to the study by (Nada, et al., 2018), they discovered that the effectiveness of the detectors falls well short of what is required in real-world scenarios. Hence, they proposed a dataset of face photos, Unconstrained Face Detection Dataset (UFDD), that include these concerns, such as weather-based degradations, focus blur, motion blur and others. This database contains an aggregation of 6,425 photos and 10,897 face annotations with major degradations or conditions which are rain, haze, lens obstructions, snow, blur, illumination changes and distractors are included.

2.6.5 Open Images Dataset V6

Open Images is a collection of 9 million images that have been annotated with image-level labels, object bounding boxes, object segmentation masks, visual connections, and localised narratives. It comprises 16 million bounding boxes for 600 item types on 1.9 million photos, making it the biggest collection with object position annotations currently available. To guarantee accuracy and uniformity, the boxes were mostly drawn by hand by skilled annotators. The photographs are diverse, and many of them feature complicated scenarios with several items (8.3 per image on average). Visual relationship annotations are also available in Open Images, showing pairings of items in certain relationships. In total, the dataset has 59.9 million image with labels covering 19,957 categories. (Google LLC, 2020)

2.7 Overview of Platforms With Free Gpu And Cpu

Google Colab and Kaggle Kernels are the two main sites that provide free GPU and CPU computational resources. Despite the fact that they are both Google products, they each have their own set of limitations and advantages. This section compares and contrasts the two system's hardware specifications, simplicity of use, and pipeline integration.

2.7.1 Hardware Specifications

Table 2.7.1(a): CPU-only VMs

Parameter	Google Colab	Kaggle Kernel
CPU Model Name	Intel(R) Xeon(R)	Intel(R) Xeon(R)
CPU Freq.	2.30GHz	2.30GHz
No. CPU Cores	2	4
CPU Family	Haswell	Haswell
Available RAM	12GB (upgradable to 26.75GB)	16GB
Disk Space	25GB	5GB

Table 2.7.1(b): GPU VMs

Parameter	Google Colab	Kaggle Kernel
GPU	Nvidia K80 / T4	Nvidia P100
GPU Memory	12GB / 16GB	16GB
GPU Memory Clock	0.82 GHz / 1.59 GHz	1.32GHz
Performance	4.1 TFLOPS / 8.1 TFLOPS	9.3 TFLOPS
Support Mixed Precision	No / Yes	No
GPU Release Year	2014 / 2018	2016
No. CPU Cores	2	2
Available RAM	12GB (upgradable to 26.75GB)	12GB
Disk Space	358GB	5GB

Table 2.7.1(c): Execution and Idle Time

Parameter	Google Colab	Kaggle Kernel
Max execution time	12 hours	9 hours
Max idle time	90 min	60 min

2.7.2 Comparison

Since 2019, Google Colab has offered two alternative GPU models: K80 and T4. However, due to availability limitations, users are unable to choose which of them they want. Each user appears to have a restricted number of access to T4 GPUs. As a result, having good GPUs in Google Colab isn't always a given.

According to Table 2.7.1(a), if the task requires more memory, Google Colab is the best option for users. Whereas , if users have a parallelizable script, Kaggle Kernel is a better option. The GPUs in Kaggle Kernels are typically better, as shown in Table 2.7.1(b). However, the T4 on Google Colab can perform mixed-precision computations. Therefore , users may benefit from float-16 training with lesser training time. Besides, Google Colab's GPUs with a high memory frequency may be able to provide superior results for RNN processes. (Kazemnejad, 2019)

Despite the fact that Colab and Kaggle provide free GPUs, the runtime is fairly short. Colab, for example, will delete the users' whole VM 12 hours after creation, regardless of how much training is left. Furthermore, users' virtual machines are deleted if they shut the Tab or disconnect from the internet for 90 minutes plus. Then they will have to rerun their experiment. This can be solved by connecting the user's VM to the Cloud storage at the beginning. Hence, they can save make a savepoint on the cloud on a regular basis. So, even if they were ejected out after 9 hours of session, they could resume the session and pick up where they left off. (Kazemnejad, 2019)

Jupyter notebooks are supported by Kaggle and Colab, however Google goes one step further and saves the notebooks to Google Drive. As a result, users may work cooperatively on their model. The service registration, particularly on the Kaggle website, is a hassle. Verification from users' phone number is required to connect the VM to the internet. Overall, Colab earns a point in this category for its simple signup process. (Kazemnejad, 2019)

2.7.3 Selection

For this project, Google Colab is chosen for data training because of its' powerful GPUs, longer execution and idle time, higher memory frequency, the availability to connect user's VM to the Cloud storage and save the checkpoint of the model on the cloud to resume their session, collaboratory accessibility and simple service registration.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Data Preparation and Annotation for Face Detection

The process of building the database can be broken into the following steps:

1. Download selected database.
2. Select 1560 images.
3. Eliminate duplicate images.
4. Apply a face detector (COCO SSD) and manually correcting false positives using ‘makesense.ai’.
5. Manually construct bounding box around the detected people and labelling them 'Human face' using ‘makesense.ai’. (Figure 3.1(a))
6. Allocate training set of 1500 photos and testing set of 300 photos (60 multiplied by 5).

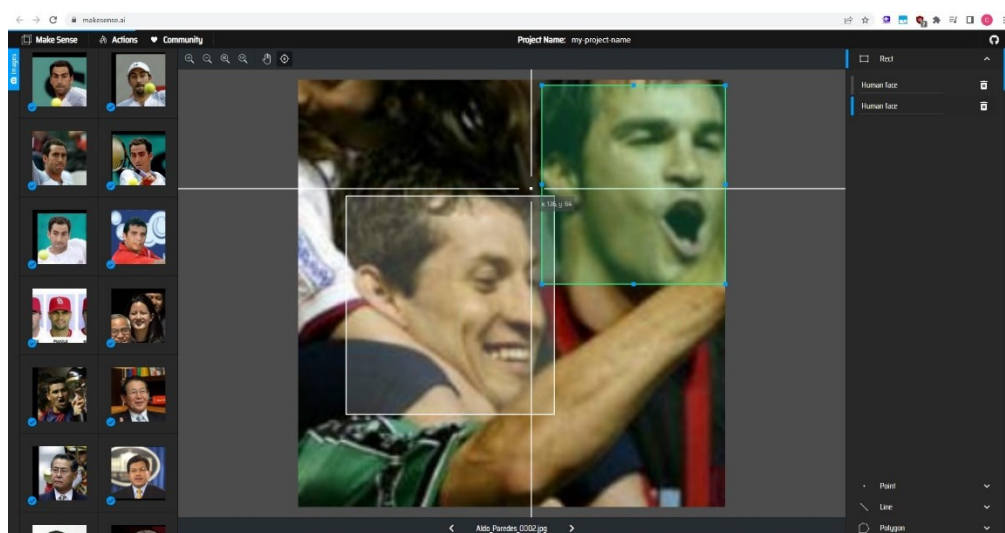


Figure 3.1(a): Constructing bounding boxes in ‘makesense.ai’

‘makesense.ai’ is a free online application for labelling images. It runs on a browser, thus, it does not require any additional installation. It is ideal for modest computer vision deep learning projects, since it simplifies and accelerates the process of building a dataset. Referring to Figure 3.1(b), labels that have been prepared can be downloaded in a variety of formats. Besides, ‘makesense.ai’ can help reduce time spent labelling images by utilising a

variety of AI models to provide users with bounding box recommendations as shown in Figure 3.1(c). (Skalski, 2019)

The AI models used are:

- SSD model, which was pre-trained on the COCO dataset, can assist users in drawing bounding boxes on images and suggesting a label.
- PoseNet model, a vision model that can determine a person's pose in an image or video by predicting the location of important bodily joints.

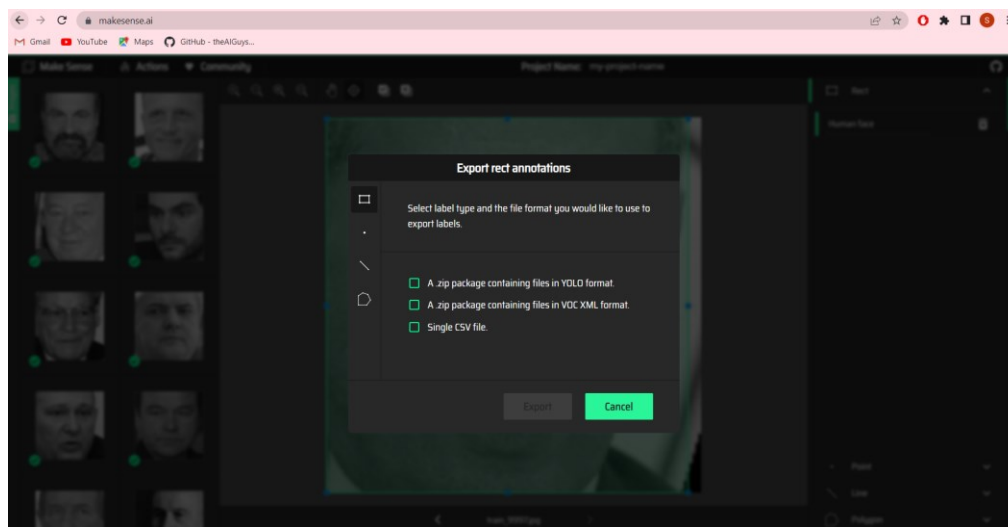


Figure 3.1(b): Annotation formats available in ‘makesense.ai’

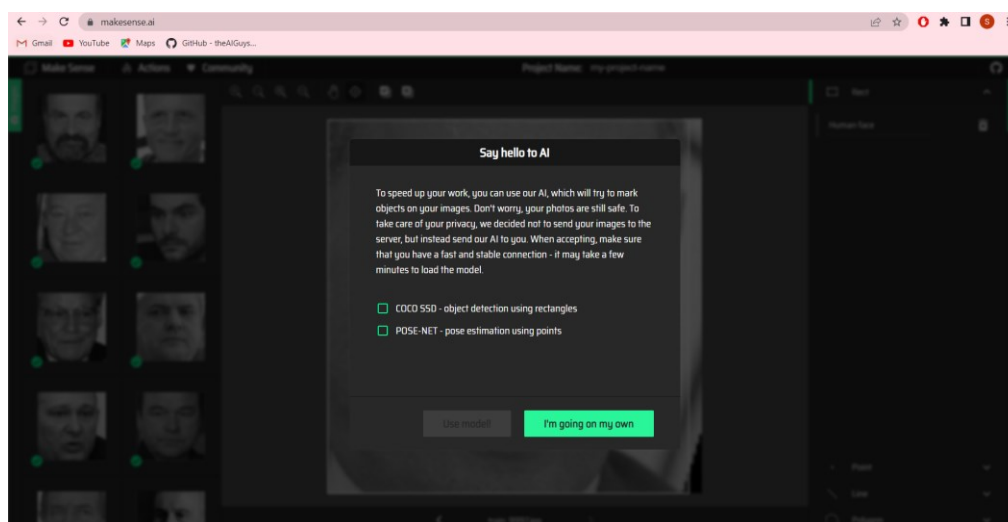


Figure 3.1(c): AI models available in ‘makesense.ai’

3.2 Data Training and Testing Protocol

In this project, there are five different training set and one testing set. Each training set consists of 1500 face images randomly picked from Labeled Faces in the Wild (LFW), Adience, Unconstrained Facial Images (UFI), Unconstrained Face Detection Dataset (UFDD) and Open Images Dataset V6, respectively. Each training set are trained for 6000 iterations. With increasing number of iterations, the accuracy of the weights also improve. The test set consists of a combination of 60 face images each from the five face database, totalling up to 300 face images.

The data in this project is divided into two sets for ‘training’ and ‘testing’. Prior to formal assessment, the training set ('obj') is used for algorithm development and general investigation. This is also known as a validation view or a model selection view. The testing set ('test'), which is for performance reporting, should only be utilised for a method's final evaluation. Before reporting, final test sets should be employed as little as possible.

3.2.1 Training Set: Selection of models and development of algorithms

The primary goal of this data view is to allow researchers to freely experiment with algorithms and parameter settings without fear of overusing test data. For example, if one is utilising support vector machines and attempting to pick which kernel to employ, it would be suitable to test multiple kernels on the training set ('obj') of the database. This view allows for the training and testing of algorithms to be performed as many times as required without appreciably skewing the final findings.

3.2.2 Testing Set: Reporting on performance

The testing set ('test') should be utilised judiciously and exclusively for performance reporting. It should ideally only be used once, because selecting the best performance from numerous algorithms or parameter settings would bias results toward artificially high accuracy. Only after selecting a model or technique (using ‘obj’ preferably), ‘test’ is used to test the performance of that algorithm.

3.3 Face Detection using YOLOv4 and YOLOv5

3.3.1 YOLOv4 Data Training

Figure 3.3.1 shows the steps to perform data training for face detection using YOLOv4 with Darknet for through Google Colab.

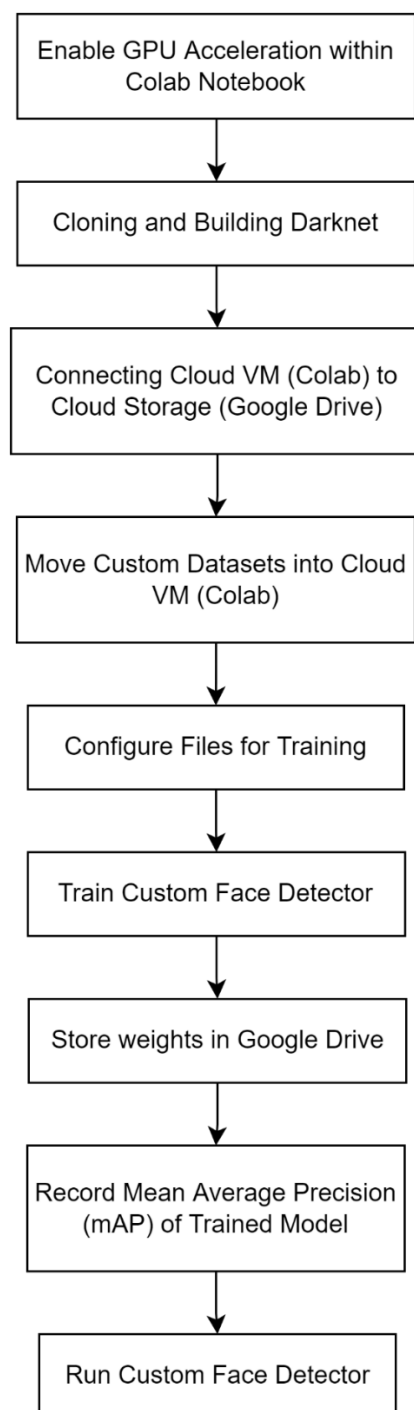


Figure 3.3.1: Block Diagram of Data Training using YOLOv4 with Darknet in Google Colab

3.3.2 Face Detection Procedure

Figure 3.3.2 shows the workflow for face detection using YOLOv4 for a static input image, video or for real-time detection.

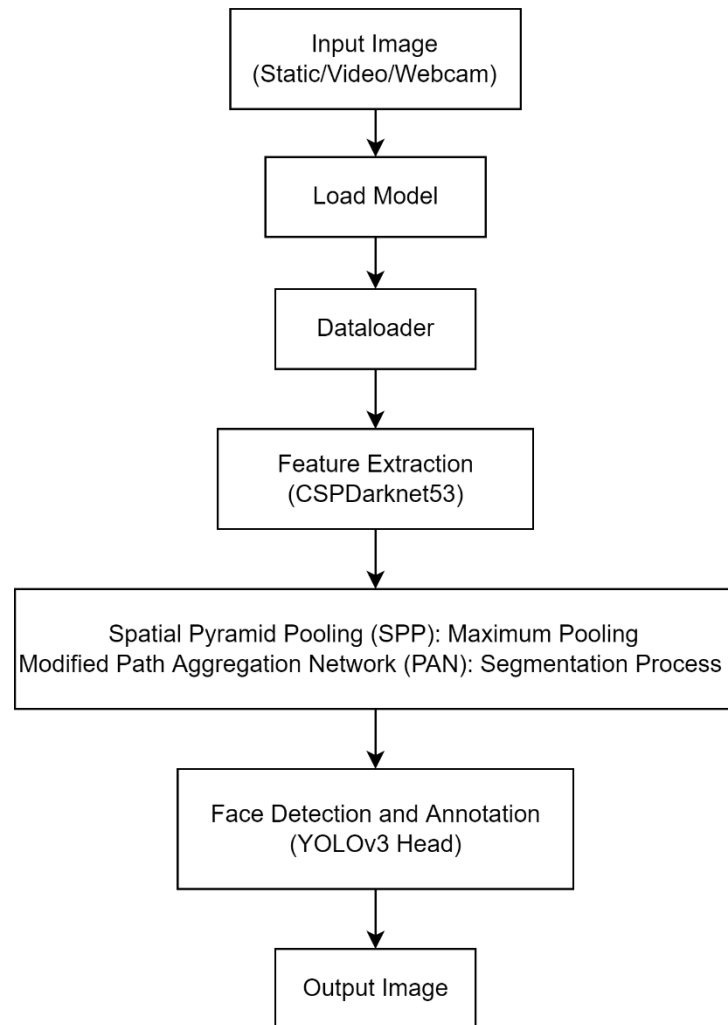


Figure 3.3.2: Face Detection Procedure using YOLOv4

3.3.3 YOLOv5 Data Training

Figure 3.3.3 shows the steps to perform data training for face detection using YOLOv5 with Pytorch locally.

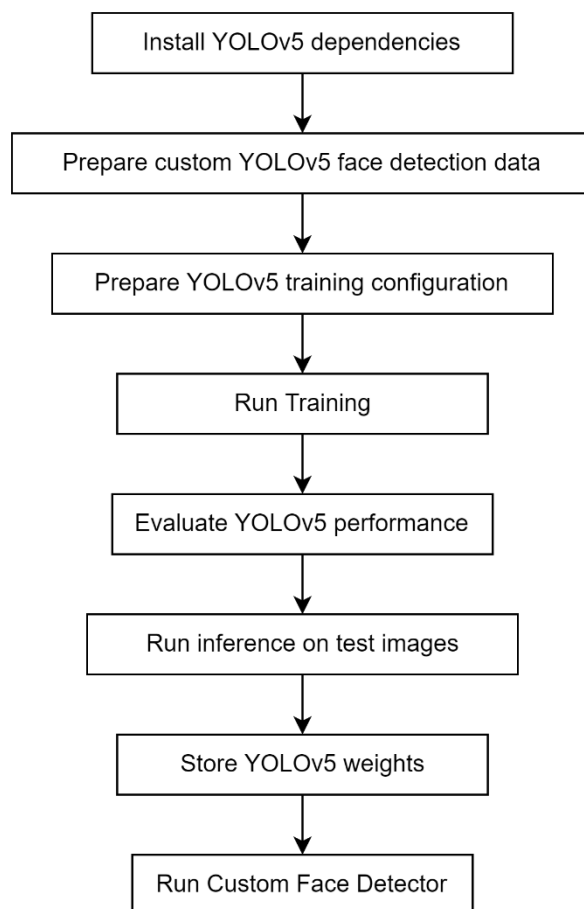


Figure 3.3.3: Block Diagram of Data Training using YOLOv5 with Pytorch on local device

3.4 Siamese Neural Network

3.4.1 Face Recognition Data Training

Figure 3.4.1 shows the steps to perform data training for face recognition using Siamese Neural Network locally.

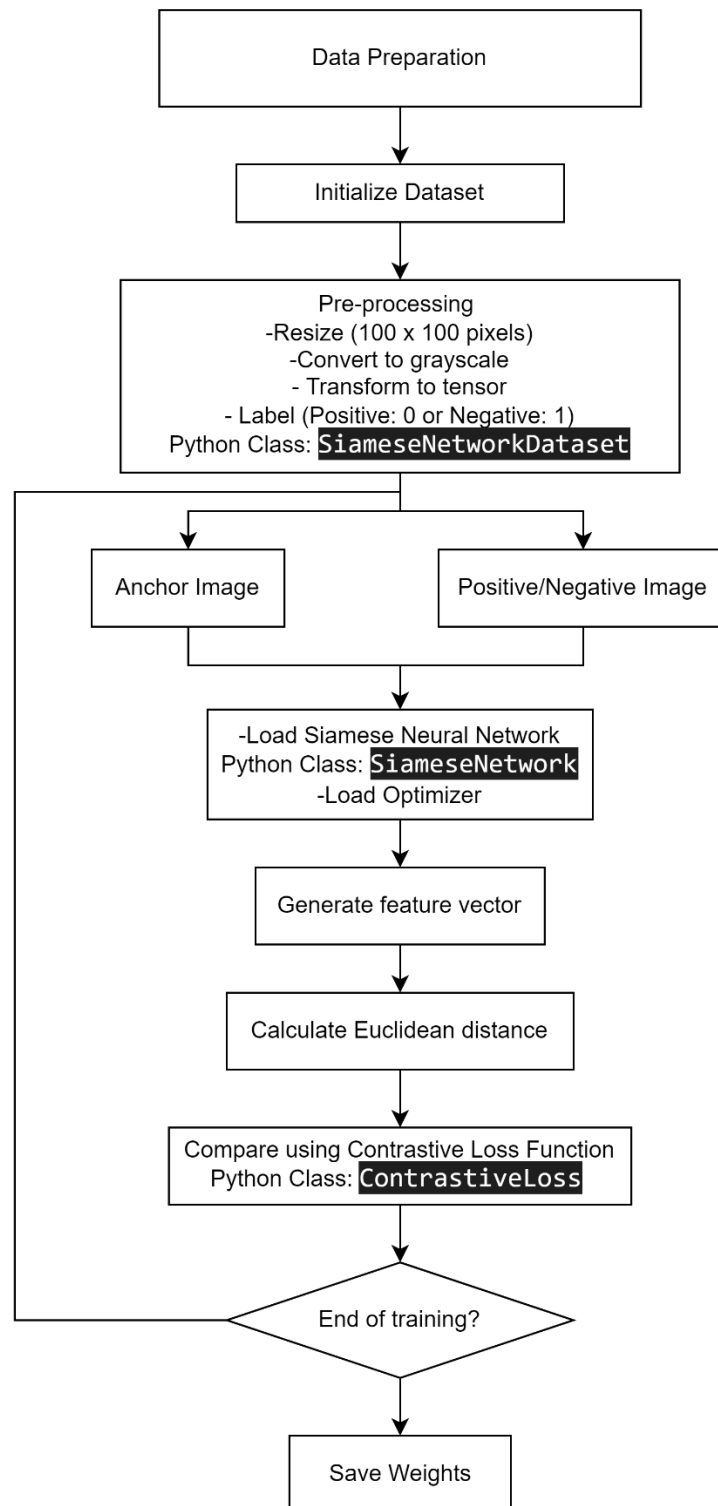


Figure 3.4.1: Siamese Neural Network Data Training

3.4.2 Face Recognition Procedure

Figure 3.4.2 shows the workflow for face recognition using Siamese Neural Network for a given input image.

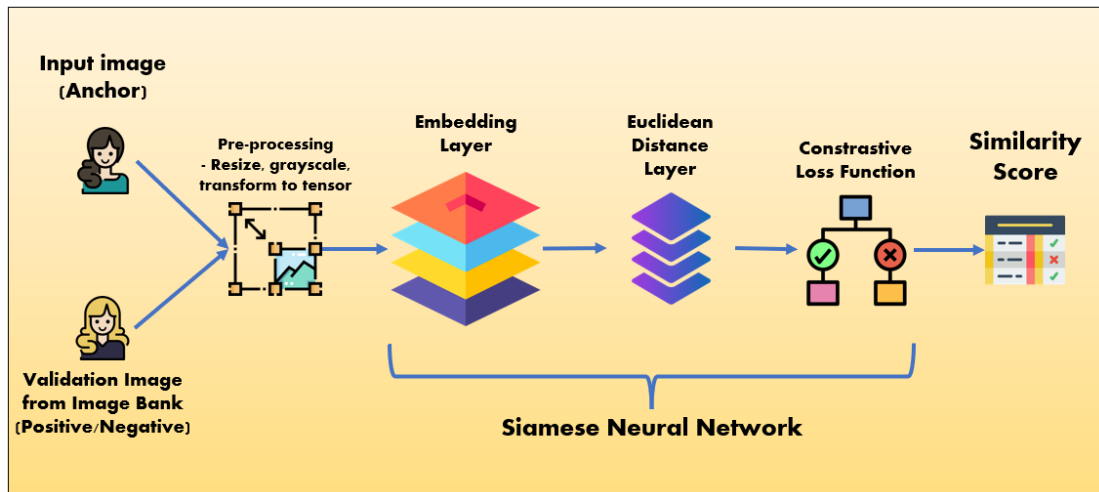


Figure 3.4.2: Face Recognition Procedure using Siamese Neural Network

3.5 Integration of Face Detection and Recognition

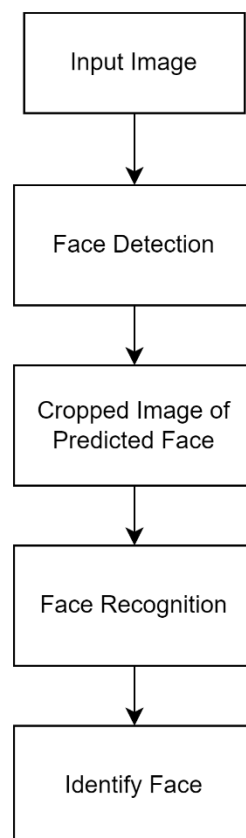


Figure 3.5: Integration of Face Detection and Recognition Model

3.6 Gantt Charts

Timeline: 25 January 2022 – 25 April 2022

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1	Discussion with Supervisor about FYP Title	25/1/2022	■													
2	Research and gather information on FYP	19/2/2022		■	■	■										
3	Software Installation, defining face detection, face recognition and unconstrained environment	19/2/2022			■	■										
4	Compare methods of face detection, test run codes and perform model training.	12/3/2022					■	■	■							
5	Use google collab to train models	16/4/2022							■	■	■	■	■	■		
6	Dataset comparison (LFW, Adience, UFI, UFDD, Open Images) and perform literature review	9/4/2022									■	■	■	■		
7	Chapter 3: Methodology, Chapter 4: Preliminary Results and preparation of presentation	25/4/2022											■	■	■	■

Timeline: 20 June 2022 – 12 September 2022

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1.	Set up weekly meeting with supervisor Discuss details of FYP 2	2022-06-20	■	■												
2.	- Perform human face training on at least 2 models - Selection of best pairing dataset and model	2022-07-04		■	■	■										
3.	- Research on human face recognition algorithms - Implement human face recognition	2022-07-25				■	■	■	■							
4.	- Development of user interface - Implementation of additional feature	2022-08-15							■	■	■	■				
5.	- Debugging and improvisation of prototype	2022-08-29										■	■	■		
6.	- Final report writing - Preparation of presentation slides	2022-09-12												■	■	■

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Evaluation Techniques

This section introduces some terms used to determine the accuracy of a trained model such as average precision (AP), mean average precision (mAP), precision (P) and recall (R).

4.1.1 Terms Involved

In the bounding box detection method, the following accuracy matrices are most frequently utilised:

1. Average Precision (AP)
2. Average Mean Precision (mAP)

The common terms used to define face detection metrics are tabulated in Table 4.1.1.

Table 4.1.1: Definition of Terms Used in Face Detection Metrics

Terms	Definition
Ground Truth (GT)	Referred to as the manually labelled bounding boxes that identifies the locations of the real objects in the picture.
True Positive (TP)	The model successfully predicted that this is a human face.
True Negative (TN)	The model successfully predicted that this is not a human face.
False Positive (FP)	Model predicts that this is a human face, but it is actually incorrect.
False Negative (FN)	Model predicts that this is not a human face, but it is actually correct.
Precision (P)	The capacity of a model to detect relevant classes, which indicates how accurate the model's prediction is. It represents the proportion of accurate positive predictions.

	<p>Defined as:</p> $P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$
Recall (R)	<p>The capacity of a model to locate all relevant classes, which indicates how accurate the model can find all the positives. It measures the proportion of accurate positives among all GT.</p> <p>Defined as:</p> $R = \frac{TP}{TP + FN} = \frac{TP}{GT}$
Average Precision (AP)	Region beneath the precision-recall curve.
Precision-Recall curve	Illustrates the trade-off between precision and recall at various thresholds.
Mean Average Precision (mAP)	Defined as the average of all classes' AP. Since there is only one class (face) in this project, AP equals mAP.

4.2 Determine End of Training

According to (AlexeyAB, 2021), 2000 iterations are adequate for each class(object), but not fewer than the amount of training photos and not less than 6000 total iterations. After multiple cycles, training should be discontinued when the average loss no longer decreases. The average loss as a consequence may be anywhere from 0.05 (small and basic model) to 3.0 (large and complex model). If training is stopped after 9000 iterations, for example, the best outcome can be obtained from one of the prior weights (7000, 8000, 9000). This occurs because of over-fitting. Over-fitting occurs when the system can recognise items on photographs from the training dataset but not on images from other datasets. Therefore, weights should be obtained from the Early Stopping Point (Figure 4.2):

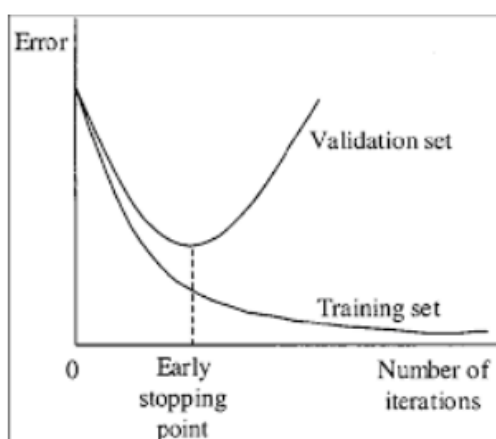


Figure 4.2: Graph showing relationship between error and number of iterations (AlexeyAB, 2021)

4.3 YOLOv4 Results

Table 4.3.1 and Table 4.3.2 below shows the results obtained from training the five datasets on YOLOv4.

Table 4.3.1: Metrics Result for YOLOv4 Data Training

Database	Open Images			Adience			LFW		
Number of iterations	P	R	mAP (%)	P	R	mAP (%)	P	R	mAP (%)
1000	0.42	0.26	25.2	0.80	0.26	30.1	0.35	0.28	22.3
2000	0.84	0.65	74.9	0.83	0.37	39.2	0.75	0.44	48.1
3000	0.81	0.77	80.4	0.72	0.40	41.8	0.86	0.44	48.9
4000	0.87	0.74	80.0	0.92	0.38	43.0	0.84	0.50	56.5
5000	0.84	0.78	81.8	0.93	0.40	44.9	0.94	0.45	53.3
6000	0.84	0.82	87.0	0.91	0.40	45.9	0.93	0.44	51.2

Table 4.3.2: Metrics Result for YOLOv4 Data Training (cont.)

Database	UFI			UFDD		
Number of iterations	P	R	mAP (%)	P	R	mAP (%)
1000	0.46	0.17	11.9	0.37	0.20	17.1
2000	0.44	0.23	21.0	0.49	0.66	62.1
3000	0.54	0.26	24.3	0.59	0.71	72.6
4000	0.46	0.26	25.0	0.82	0.69	77.3
5000	0.45	0.25	21.7	0.74	0.75	80.8
6000	0.37	0.24	22.2	0.78	0.75	81.6

Table 4.3.3: Highest mAP achieved using YOLOv4

Databases	Number of iterations	Highest mAP (%)
Open Images	6000	87.0
Adience	6000	45.9
LFW	4000	56.5
UFI	4000	25.0
UFDD	6000	81.6

Discussion: Referring to Table 4.3.1, Table 4.3.2 and Graph A-1 in Appendix A, the overall trend for all databases shows that the mean average precision (mAP) increases with increasing number of iterations. The precision and recall also increases with greater iterations. The highest performing database is Open Images (87% mAP for 6000 iterations) followed by UFDD, LFW, Adience and lastly UFI according to Table 4.3.3. LFW and UFI reaches their highest mAP at 4000 iterations and becomes over-fitted. Hence, the best weight should be chosen when the weight is generated at 4000 iterations for LFW and UFI. The best weight for the other datasets, Open Images, Adience and UFDD, should be taken during the 6000 iterations.

4.4 YOLOv5 Results

Table 4.4.1 and Table 4.4.2 below shows the results obtained from training the five datasets on YOLOv5.

Table 4.4.1: Metrics Result for YOLOv5 Data Training

Database	Open Images			Adience			LFW		
Number of iterations	P	R	mAP (%)	P	R	mAP (%)	P	R	mAP (%)
1000	0.805	0.550	63.5	0.616	0.285	30.5	0.657	0.335	36.0
2000	0.799	0.751	81.4	0.803	0.472	53.9	0.847	0.452	51.8
3000	0.911	0.747	84.9	0.792	0.481	54.4	0.855	0.460	54.1
4000	0.897	0.753	84.1	0.878	0.512	60.1	0.898	0.488	58.7
5000	0.920	0.763	86.1	0.828	0.552	61.3	0.819	0.557	62.9
6000	0.870	0.777	84.1	0.872	0.509	60.8	0.808	0.540	61.3

Table 4.4.2: Metrics Result for YOLOv5 Data Training (cont.)

Database	UFI			UFDD		
Number of iterations	P	R	mAP (%)	P	R	mAP (%)
1000	0.382	0.195	13.7	0.892	0.602	71.2
2000	0.545	0.239	22.0	0.879	0.683	79.2
3000	0.556	0.242	22.5	0.904	0.730	85.3
4000	0.555	0.268	24.2	0.864	0.772	84.9
5000	0.700	0.299	29.4	0.887	0.749	84.1
6000	0.684	0.289	28.1	0.874	0.782	85.4

Table 4.4.3: Highest mAP achieved using YOLOv5

Databases	Number of iterations	Highest mAP (%)
Open Images	5000	86.1
Adience	5000	61.3
LFW	5000	62.9
UFI	5000	29.4
UFDD	6000	85.4

Discussion: Referring to Table 4.4.1, Table 4.4.2 and Graph A-2 in Appendix A, the overall trend for all databases shows that the mean average precision (mAP) increases with increasing number of iterations. The precision and recall also increases with greater iterations. The highest performing database is Open Images (86.1% mAP for 5000 iterations) followed by UFDD, LFW, Adience and lastly UFI according to Table 4.4.3. All the database except UFDD reach their highest mAP at 5000 iterations. Hence, the best weight should be selected when the weight is generated at 5000 iterations for all the database except UFDD.

4.5 Additional YOLOv4 and YOLOv5 Results

Table 4.5.1: Train, test time and weights size for YOLOv4 and YOLOv5

Database	Open Images		Adience		LFW	
	YOLOv5	YOLOv4	YOLOv5	YOLOv4	YOLOv5	YOLOv4
Training time (hours)	0.829	7.2	0.789	8.5	0.833	8.3
Testing time for 300 test images (s)	34.8	113.0	34.5	114.0	29.9	113.0
Weights size (MB)	14.7	244	14.7	244	14.7	244

Table 4.5.2: Train, test time and weights size for YOLOv4 and YOLOv5 (cont.)

Database	UFI		UFDD	
Model	YOLOv5	YOLOv4	YOLOv5	YOLOv4
Training time (hours)	0.828	7.8	1.547	8.1
Testing time for 300 test images (s)	35.7	122.0	60.0	34.0
Weights size (MB)	14.7	244	14.7	244

Discussion: Referring to Table 4.5.1 and Table 4.5.2, the training time for 1500 images for 6000 iterations on YOLOv4 takes roughly around 8 hours of non-stop operation whereas it only takes roughly an hour to generate the weights file for YOLOv5. Moreover, the testing time for 300 test images on YOLOv4 takes roughly around 30 seconds to 2 minutes while YOLOv5 requires about 30 seconds to a minute. Additionally, YOLOv5 produces a much smaller weight file of 14 MB compared to YOLOv4 of 244 MB.

4.6 Comparison of YOLOv4 and YOLOv5 Results

It was found that YOLOv5 perform similarly with YOLOv4 using the mAP metrics. However, the training time of 6000 iterations for YOLOv5 is considerably shorter than YOLOv4. YOLOv4 needs roughly 8 hours of non-stop operation, however YOLOv5 only needed an hour. YOLOv5 is also able to perform face detection on 300 test images within a shorter period. Additionally, YOLOv5 produces a much smaller weight file of 14 MB as opposed to YOLOv4's 244 MB weight file. Hence, it can be deduced that YOLOv5 has greater advantages although it's performance is almost the same to YOLOv4. The YOLOv5 model performed best using the Open Images dataset (86.1% mAP).

4.7 Face Detection Result

Table 4.7.1 shows some face detection result using YOLOv5 on images taken in varying unconstrained environment for all the databases.

Table 4.7.1: YOLOv5 Face Detection Result for Open Images Database and Adience


Unconstrained Environment	Open Images	Adience
Rain		
Snow		
Haze		
Blur		
Illumination		
Lens impediment		

Table 4.7.2: YOLOv5 Face Detection Result for UFDD, LFW and UFI

Unconstrained Environment	Database		
	UFDD	LFW	UFI
Rain	A child playing in a water fountain with rain. Three faces are detected with bounding boxes and confidence scores: 0.94, 0.95, and 0.94.	A child playing in a water fountain with rain. Two faces are detected with bounding boxes and confidence scores: 0.94 and 0.85.	A child playing in a water fountain with rain. No bounding boxes or confidence scores are visible.
Snow	A person in winter gear in a snowy environment. One face is detected with a bounding box and confidence score of 0.94.	A person in winter gear in a snowy environment. One face is detected with a bounding box and confidence score of 0.95.	A person in winter gear in a snowy environment. No bounding boxes or confidence scores are visible.
Haze	A group of people in a hazy environment. Two faces are detected with bounding boxes and confidence scores: 0.92 and 0.94.	A group of people in a hazy environment. Two faces are detected with bounding boxes and confidence scores: 0.31 and 0.82.	A group of people in a hazy environment. No bounding boxes or confidence scores are visible.
Blur	A person on a treadmill in a blurred gym setting. Three faces are detected with bounding boxes and confidence scores: 0.94, 0.70, and 0.67.	A person on a treadmill in a blurred gym setting. One face is detected with a bounding box and confidence score of 0.93.	A person on a treadmill in a blurred gym setting. No bounding boxes or confidence scores are visible.
Illumination	A person's face in a bright, high-contrast lighting environment. One face is detected with a bounding box and confidence score of 0.93.	A person's face in a bright, high-contrast lighting environment. One face is detected with a bounding box and confidence score of 0.39.	A person's face in a bright, high-contrast lighting environment. No bounding boxes or confidence scores are visible.
Lens impediment	A person's face seen through a lens with water droplets. One face is detected with a bounding box and confidence score of 0.94.	A person's face seen through a lens with water droplets. One face is detected with a bounding box and confidence score of 0.31.	A person's face seen through a lens with water droplets. No bounding boxes or confidence scores are visible.

Discussion: Referring to Table 4.7.1 and Table 4.7.2, Open Images is able to detect all the faces, with high accuracy, in the six images taken in the unconstrained environment of rain, snow, haze, blur, illumination and lens impediment accurately with no false positives. UFDD and LFW are also able to detect the faces in the six images with high accuracy but with some false positives. Adience can also detect some faces but with a lower accuracy and some false positives. Lastly, UFI performed the worst and is unable to detect any faces in all the six images.

4.8 Result for Face Recognition using Siamese Neural Network

The LFW dataset is used to train the Siamese Neural Network for face recognition. LFW(40) represents training data of 40 individuals with 10 images respectively. LFW(1573) is the full dataset of LFW with 1573 individuals with varying images. LFW(40) is trained for 300 epochs until the loss curve saturates around 0 loss. Whereas, LFW(1573) is trained for 100 epochs until the loss curve saturates around 0.25.

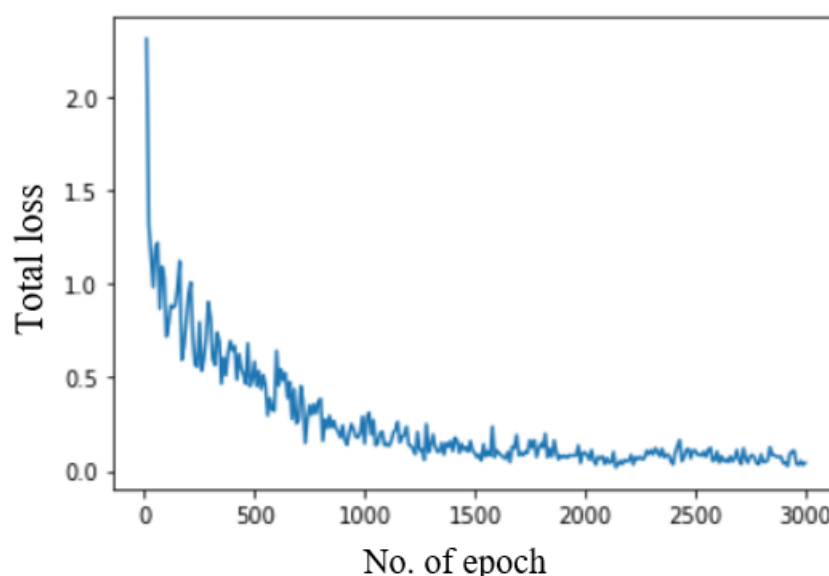


Figure 4.8.1: Graph of total loss vs no. of epoch for 300 epochs for LFW(40)

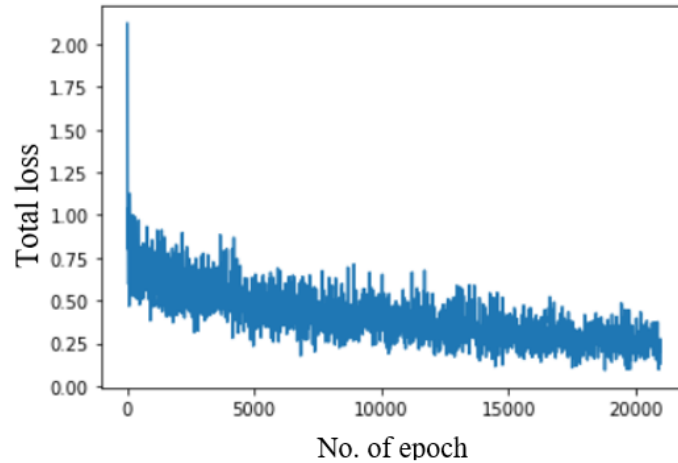
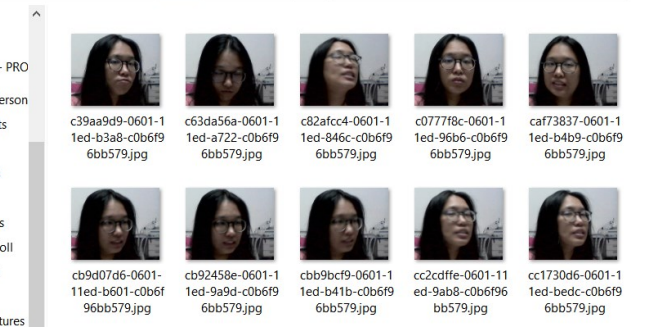





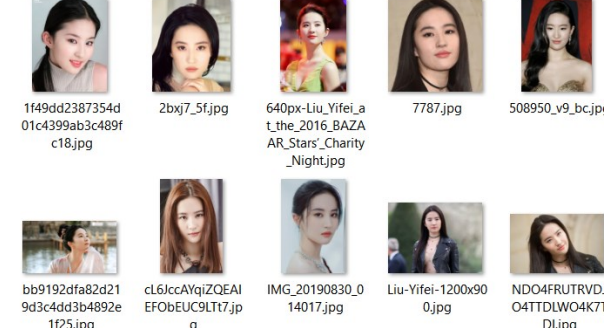



Figure 4.8.2: Graph of total loss vs no. of epoch for 100 epochs for LFW(1573)

Referring to Table 4.8.1, a test folder consisting of 10 individuals with 10 images respectively is verified with a given input image (anchor). The total images of this test folder is 100 images. The anchor is an image that has never appeared in the dataset before. The Siamese Neural Network will compare the anchor with all the images in the test folder continuously and compute the Euclidean distance. A threshold of 1 is used. A low 'similarity score' indicates that the anchor is most likely the same person as in the person in the image compared. The 10 individuals are 'Ariana', 'Bill_Clinton', 'Britney_Spears', 'Gordon_Brown', 'Kanye_West', 'Laura_Bush', 'Liu_Yifei', 'Mahathir_Mohammad', 'Siti_Nurhaliza' and 'Tom_Holland' as shown in Table 4.8.1. The individuals 'Bill_Clinton', 'Britney_Spears', 'Gordon_Brown' and 'Laura_Bush' are known individuals that were trained in the LFW dataset whereas the other individuals are totally new unknown faces to the Siamese Neural Network model.

Table 4.8.1: 10 verification images of 10 individuals for comparison

Person	Verification Images
Ariana	<p>« verification_images > Ariana <input type="text" value="Search Ariana"/></p>  <p>Grid of 10 verification images for Ariana Grande. Each image is accompanied by a unique alphanumeric ID and a .jpg file extension.</p>
Bill_Clinton	<p>« verification_images > Bill_Clinton <input type="text" value="Search Bill_Clinton"/></p>  <p>Grid of 10 verification images for Bill Clinton. Each image is accompanied by a unique alphanumeric ID and a .jpg file extension.</p>
Britney_Spears	<p>« verification_im... > Britney Spears <input type="text" value="Search Britney Spears"/></p>  <p>Grid of 10 verification images for Britney Spears. Each image is accompanied by a unique alphanumeric ID and a .jpg file extension.</p>
Gordon_Brown	<p>verification_i... > Gordon_Brown <input type="text" value="Search Gordon_Brown"/></p>  <p>Grid of 10 verification images for Gordon Brown. Each image is accompanied by a unique alphanumeric ID and a .jpg file extension.</p>

<p>Kanye_West</p>	<p>verification_imag... > Kanye_West</p> <p>Search Kanye_West</p>  <p>Grid of 10 images of Kanye West with various captions including file names and identifiers.</p>
<p>Laura_Bush</p>	<p>verification_images > Laura_Bush</p> <p>Search Laura_Bush</p>  <p>Grid of 10 images of Laura Bush with captions like Laura_Bush_0001.jpg through Laura_Bush_0010.jpg.</p>
<p>Liu_Yifei</p>	<p>verification_images > Liu_Yifei</p> <p>Search Liu_Yifei</p>  <p>Grid of 10 images of Liu Yifei with captions including file names and event references like '2016_BAZAAR_Stars_Charity_Night'.</p>
<p>Mahathir_Mohammad</p>	<p>verificati... > Mahathir_Mohamad</p> <p>Search Mahathir_Mohamad</p>  <p>Grid of 10 images of Mahathir Mohammad with captions like Mahathir_Mohamad_0002.jpg through Mahathir_Mohamad_0013.jpg.</p>

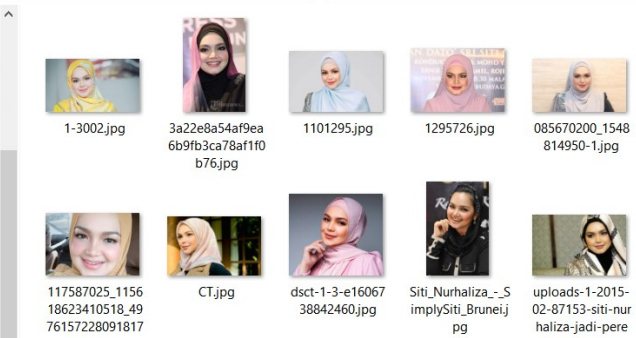
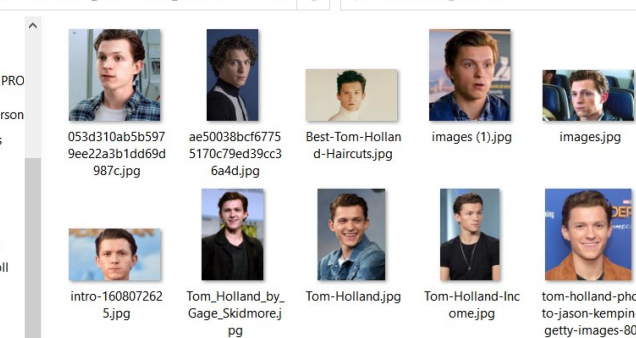


<p>Siti_Nurhaliza</p>	<p>verification_im... > Siti_Nurhaliza</p> <p>Search Siti_Nurhaliza</p>  <p>1-3002.jpg 3a22e8a54af9ea6b9fb3ca78af1f0b76.jpg 1101295.jpg 1295726.jpg 085670200_1548814950-1.jpg</p> <p>117587025_115618623410518_4976157228091817579_n.jpg CT.jpg dsct-1-3-e1606738842460.jpg Siti_Nurhaliza_-_SimplifySiti_Brunei.jpg uploads-1-2015-02-87153-siti-nurhaliza-jadi-perempuan-terkaya...</p>
<p>Tom_Holland</p>	<p>verification_ima... > Tom_Holland</p> <p>Search Tom_Holland</p>  <p>053d310ab5b5979ee22a3b1dd69d987c.jpg ae50038bc67755170c79ed39cc36a4d.jpg Best-Tom-Holland-Haircuts.jpg images (1).jpg images.jpg</p> <p>intro-1608072625.jpg Tom_Holland_by_Gage_Skidmore.jpg Tom-Holland.jpg Tom-Holland-Income.jpg tom-holland-photo-jason-kempin-getty-images-801510482-profile...</p>

Table 4.8.2: Results obtained from known and unknown faces in the dataset
using Siamese Neural Network

Input Image (Anchor)	Dataset	Face	Total Time (s)	Top Identified Names	Average Similarity Score
 Bill_Clinton	LFW (1573)	Known	4.783	Bill_Clinton Mahathir_Mohamad Kanye_West	0.421 2.477 2.602
	LFW (40)	Known	5.030	Bill_Clinton Siti_Nurhaliza Britney Spears	1.126 4.176 4.441
 Britney_Spears	LFW (1573)	Known	5.283	Britney_Spears Liu_Yifei Ariana	0.441 1.741 2.002
	LFW (40)	Known	5.326	Ariana Britney_Spears Tom_Holland	2.131 2.401 3.324
 Gordon_Brown	LFW (1573)	Known	5.951	Kanye_West Siti_Nurhaliza Mahathir_Mohamad	1.148 1.599 1.660
	LFW (40)	Known	4.874	Mahathir_Mohamad Siti_Nurhaliza Tom_Holland	2.260 2.542 2.589
 Liu_Yifei	LFW (1573)	Unknown	5.037	Britney_Spears Liu_Yifei Ariana	0.667 1.517 1.841
	LFW (40)	Unknown	4.985	Mahathir_Mohamad Kanye_West Siti_Nurhaliza	1.868 1.875 1.982
 Kanye_West	LFW (1573)	Unknown	5.135	Britney_Spears Siti_Nurhaliza Kanye_West	0.939 1.653 2.252
	LFW (40)	Unknown	4.782	Britney_Spears Kanye_West' Siti_Nurhaliza	1.362 1.484 1.961
 Mahathir_Mohamad	LFW (1573)	Unknown	5.224	Bill_Clinton Mahathir_Mohamad Siti_Nurhaliza	0.803 2.024 2.924
	LFW (40)	Unknown	5.448	Bill_Clinton Siti_Nurhaliza Britney Spears	2.289 3.356 3.195

Discussion: Referring to Table 4.8.2, generally, the similarity score between the input image and the verification image is much lesser for LFW(1573) compared to LFW(40). The Siamese Neural Network takes about 4-6 seconds to identify the faces whether the input image is a known or unknown face. The LFW(1573) seems to be able to identify all known and unknown face images more accurately and consistently compared to LFW(40). Although there are similar matching verification individuals, the similarity score between the anchor and the correct verification individual is quite low. Out of three known images, 'Bill_Clinton', 'Britney_Spears' and 'Gordon_Brown', the LFW(1573) was able to identify two of them, 'Bill_Clinton' and 'Britney_Spears'. Additionally, the LFW(1573) was also able to identify the unknown input faces 'Liu_Yifei', 'Kanye_West' and 'Mahathir_Mohammad' under the top three searches. Hence, it can be deduced that the Siamese Neural Network is most likely to perform better with larger database with more labelled individuals.

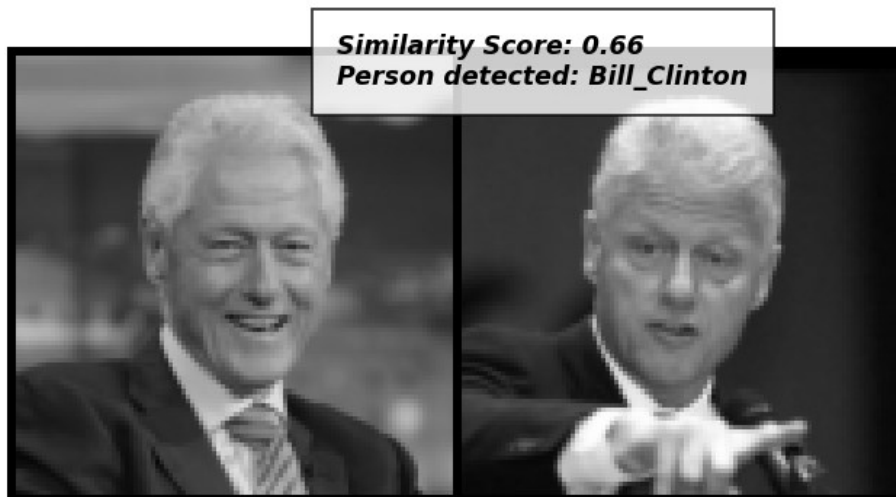


Figure 4.8.3: Face similarity on known face (Bill_Clinton)



Figure 4.8.4: Face similarity on known face (Britney_Spears)

Discussion: Figure 4.8.3 and Figure 4.8.4 shows a sample verification process of the Siamese Neural Network to confirm the individual's identity from the given input image. Figure 4.8.3 has a similarity score of 0.66 between the anchor (left) and verification image (right). Using a threshold of 1, the similarity score is less than the threshold and this indicates that the person in the anchor and verification image are the same person. The name 'Bill_Clinton', is retrieved from the labelled folder's name that stores the verification image. Therefore, the two images belong to the person 'Bill_Clinton'. The same concept applies to Figure 4.8.4.

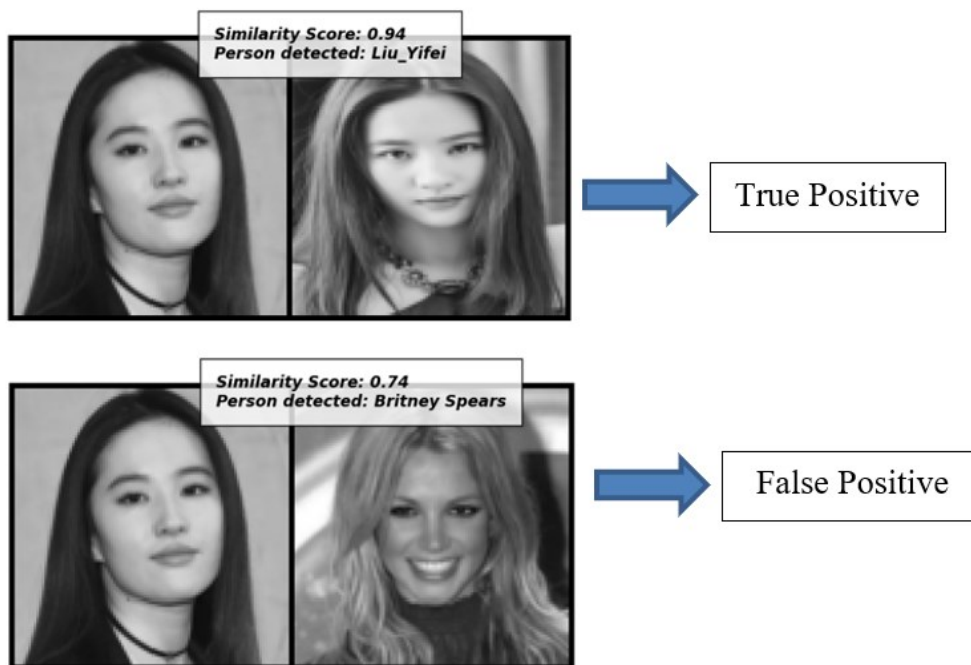


Figure 4.8.5: Face similarity on unknown face (Liu_Yifei)

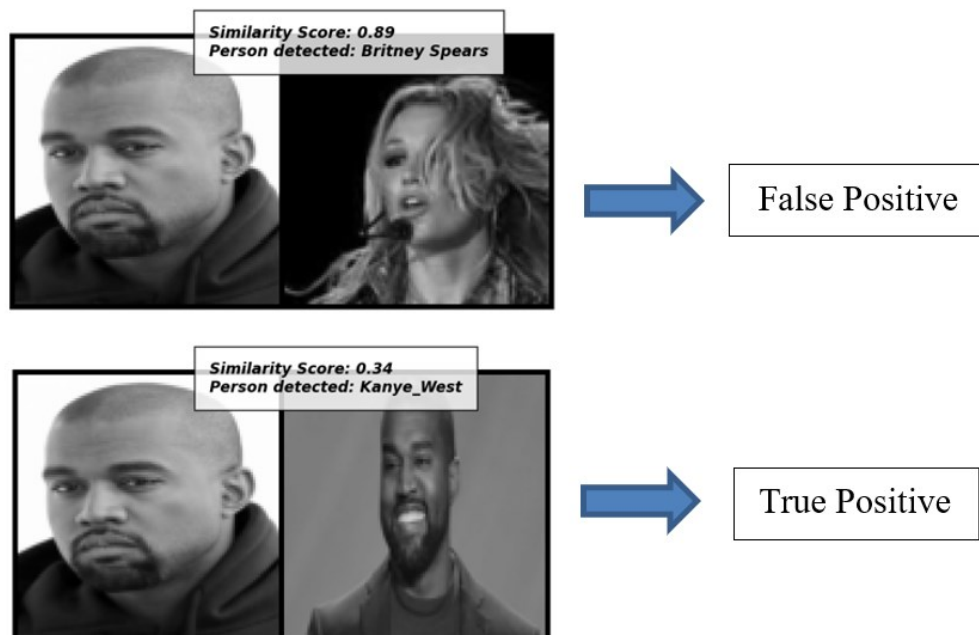


Figure 4.8.6: Face similarity on unknown face (Kanye_West)

Discussion: Figure 4.8.5 and Figure 4.8.6 shows that the Siamese Neural Network can have multiple matching individuals for an input image. This happens when the two images have similar facial features which results to a similar feature vector. If the person in the input image is matched with a wrong person but the model considers it correct, it is known as a false positive. It is only considered a true positive if the model matches the input image with the correct individual. Hence, the Siamese Neural Network can be further improved by increasing the number of verification images for each individual to reduce the likelihood of the model predicting a false positive.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Despite the abundance of solutions and the sincere efforts of many academics, face detection and recognition remains a difficult subject. In this project, five face datasets—Labeled Faces in the Wild (LFW), Adience, Unconstrained Facial Images (UFI), Open Images V6 and Unconstrained Face Detection Dataset (UFDD)—are manually annotated and trained on models YOLOv4 and YOLOv5 in an uncontrolled environment. The training set consists of 1500 images, and the test set is composed of 300 images (includes rain, snow, haze, blur, illumination and lens impediments). The LFW dataset is then trained using single-shot face recognition on Siamese neural networks. The mAP measurements showed that YOLOv5 and YOLOv4 performed equally. However, YOLOv5's training period of 6000 iterations is considerably shorter than YOLOv4's. Additionally, YOLOv5 generates a file that is 14 MB in size, which is substantially lower than YOLOv4. For 300 test photographs shot in an unrestricted environment, the YOLOv5 model performed best using the Open Images dataset (86.1% mAP). When Siamese Neural Networks were used to verify the result with known or unknown faces in the database, the bigger dataset LFW with more tagged people (1573 individuals) was capable of delivering satisfactory results. Adding more verification photographs for each person will help the Siamese Neural Network perform even better by decreasing the possibility of a false positive prediction.

5.2 Recommendations for future work

The performance of the face detection model using YOLOv5 can be further studied by using a larger training data size, test size and number of training iterations. Although the face detection model performs with a satisfactory performance in an unconstrained environment, the impact of varying degree of the unconstrained environment have not been studied. For example, the impact of varying illumination degree and how bright or dark can the face detection model tolerate and retain its performance. Future work may include investigating how well the model works depending on the severity of the image quality in an unconstrained environment. Although the Siamese Neural Network for face recognition is fast, it may produce multiple positive match since it relies on similarity score instead of classifying the human face features. For a small dataset of 40 individuals with 10 images of each individuals, the Siamese Neural Network often fails to identify the face when a new individual is introduced. Hence, the performance of the face recognition model can be further studied by using a larger dataset of individuals for training and testing and a greater number of training iterations. Additionally, alternative face recognition approach that performs pair matching may also be studied to compare the performance of the face identification process. Other future works include implementation of the face detection and recognition system on other platforms such as Android. A drawback of the YOLOv5 model is its inability to be implemented on Tensorflow Lite for android development since it is not supported. Hence, other approach to implement the model may be studied further.

REFERENCES

- AlexeyAB. (2021). *YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)*. Retrieved April 23, 2022, from <https://github.com/AlexeyAB/darknet#how-to-mark-bounded-boxes-of-objects-and-create-annotation-files>
- Balcazar, C. (2020, November 29). *How Augmented Reality lets Sephora “try on” something different*. Retrieved April 7, 2022, from <https://medium.com/marketing-in-the-age-of-digital/how-augmented-reality-lets-sephora-try-on-something-different-23b4446fd5c1>
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *Computer Vision and Pattern Recognition*.
- Garg, A. (2021, December 14). *How to Use Yolo v5 Object Detection Algorithm for Custom Object Detection*. Retrieved September 8, 2022, from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2021/12/how-to-use-yolo-v5-object-detection-algorithm-for-custom-object-detection-an-example-use-case/#:~:text=It%20is%20a%20novel%20convolutional,and%20probabilities%20for%20each%20component>.
- Gargaro, D. (2021, July 20). *The pros and cons of facial recognition technology*. Retrieved April 7, 2022, from <https://www.itpro.com/security/privacy/356882/the-pros-and-cons-of-facial-recognition-technology>
- Google LLC. (2020). *Overview of Open Images V6*. Retrieved April 20, 2022, from <https://storage.googleapis.com/openimages/web/factsfigures.html>
- Grudin, M. (1997). A compact multi-level model for the recognition of facial images. *Liverpool John Moores Univ.*
- Hoo, S. C., & Ibrahim, H. (2019). Biometric-Based Attendance Tracking System for Education Sectors: A Literature Survey on Hardware Requirements. *Journal of Sensors, 2019*, 1-25.

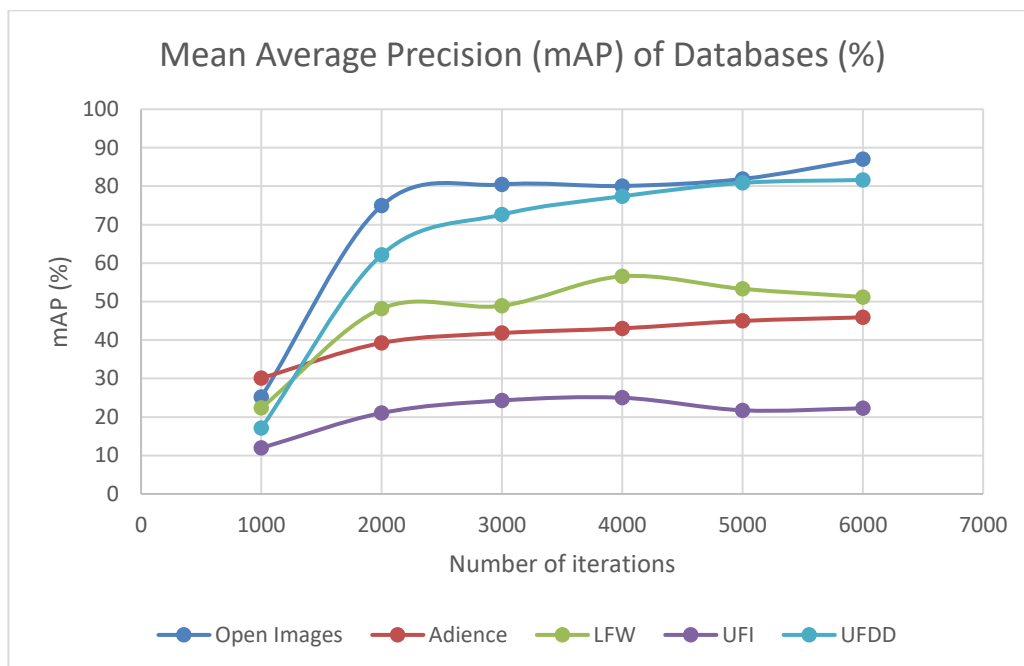
- Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007, October). Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. *University of Massachusetts, Amherst, Technical Report*, 7-49.
- kaspersky. (n.d.). *What is Facial Recognition – Definition and Explanation*. Retrieved April 20, 2022, from <https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition>
- Kazemnejad, A. (2019). *How to do Deep Learning research with absolutely no GPUs - Part 2*. Retrieved April 20, 2022, from https://kazemnejad.com/blog/how_to_do_deep_learning_research_with_absolutely_no_gpus_part_2/
- Lawrence, S., Giles, C., Tsoi, A., & Back, A. (1997). Face recognition: A convolutional neural-network approach. *IEEE Trans. Neural Networks*, 8, 98-113.
- Lenc, L., & Kral, P. (2015, October 25-31). Unconstrained Facial Images: Database for Face Recognition under Real-world Conditions. *14th Mexican International Conference on Artificial Intelligence (MICAI 2015)*.
- Lin, S., Kung, S., & Lin, L. J. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans. Neural Networks*, 8, 114-132.
- Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.
- Minaee, S., Luo, P., Lin, Z., & Bowyer, K. (2021, March 27). Going Deeper Into Face Detection: A Survey. *arXiv:2103.14983*, 1, 1-17.
- Nada, H., Sindagi, V. A., Zhang, H., & Patel, V. M. (2018). Pushing the Limits of Unconstrained Face Detection: a Challenge Dataset and Baseline Results. *arXiv preprint arXiv:1804.10275*, 3, 1-10.
- Protik, A. A., Rafi, A. H., & Siddique, S. (2021). Real-time Personal Protective Equipment (PPE) Detection Using YOLOv4 and TensorFlow. *Conference: IEEE Region 10 Symposium (TENSymp)*. doi:10.1109/TENSymp52854.2021.9550808

- Samaria, F., & Harter, A. (1994). Parameterisation of a stochastic model for human face identification. *Proc. Second IEEE Workshop Applications of Computer Vision*.
- Samaria, Ferdinando, & Fallside. (1994). Face Identification and Feature Extraction Using Hidden Markov Models. *Elsevier*.
- Shepley, A. J. (2019, July 12). Face Recognition in Unconstrained Conditions: A Systematic Review. *arXiv*, 1, 1-19.
- Sirovich, M. K. (1990). Application of the Karhunen-Loève procedure for the characterisation of human faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12, 831-835.
- Skalski, P. (2019). *makesense.ai*. Retrieved April 21, 2022, from <https://skalskip.github.io/make-sense/>
- Solawetz, J. (2020, June 29). *YOLOv5 New Version - Improvements And Evaluation*. Retrieved September 9, 2022, from [roboflow.com: https://blog.roboflow.com/yolov5-improvements-and-evaluation/#:~:text=The%20YOLOv5%20repository%20is%20a,then%20move%20forward%20to%20production.](https://blog.roboflow.com/yolov5-improvements-and-evaluation/#:~:text=The%20YOLOv5%20repository%20is%20a,then%20move%20forward%20to%20production.)
- Stefanovic, S. (2021, November 7). # 019 Siamese Network in PyTorch with application to face similarity. Retrieved September 10, 2022, from Datahacker: <https://datahacker.rs/019-siamese-network-in-pytorch-with-application-to-face-similarity/#contrastive-loss-function>
- The Open University of Israel and Adience. (2014). *Unfiltered faces for gender and age classification*. Retrieved April 6, 2022, from <https://talhassner.github.io/home/projects/Adience/Adience-data.html>
- Tolba, A. S., El-Baz, A., & El-Harby, A. (2006). Face Recognition: A Literature Review. *International Journal of Signal Processing*, 1-16.
- Truein. (2021, October 25). *8 Advantages of Using Face Recognition Attendance System At Workplace*. Retrieved April 7, 2022, from <https://truein.com/advantages-of-face-recognition-attendance-system/>
- Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). *WIDER FACE: A Face Detection Benchmark*. Retrieved April 5, 2022, from [shuoyang1213.me: http://shuoyang1213.me/WIDERFACE/](http://shuoyang1213.me/WIDERFACE/)

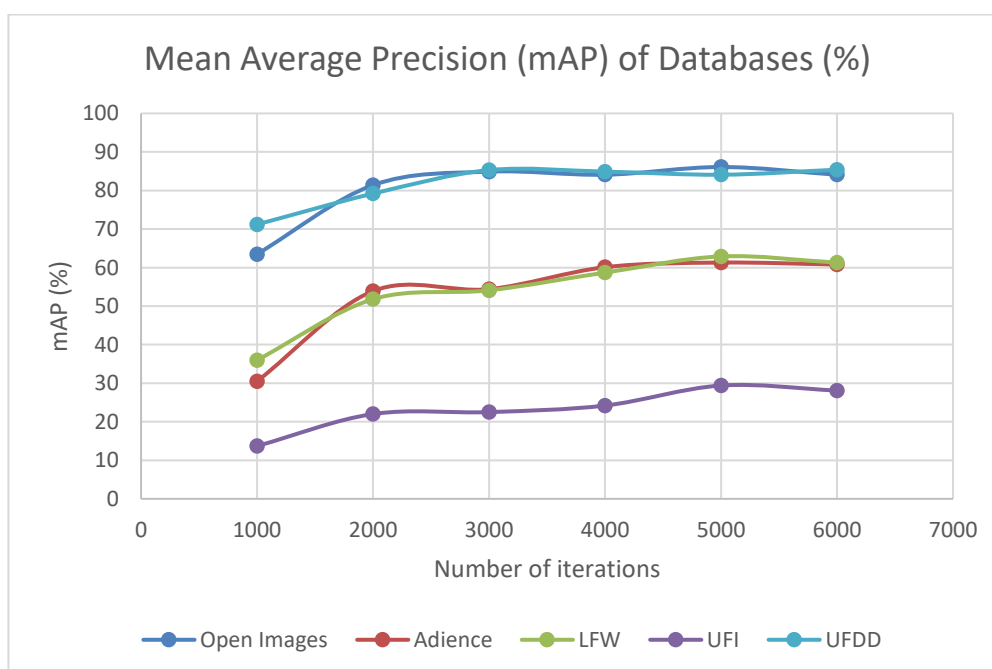
Zheng, L. (2000). A new model-based lighting normalization algorithm and its application in face recognition. *National University of Singapore*.

APPENDICES

Appendix A: Graphs



Graph A-1: Mean Average Precision (mAP) vs Number of Iterations for YOLOv4



Graph A-2: Mean Average Precision (mAP) vs Number of Iterations for YOLOv5

Appendix B: Tables

Table B-1: Output of YOLOv4 data training for Open Images Dataset V6

Number of iterations	Output
1000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_1000.weights... seen 64, trained: 64 K-images (1 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 18178, unique_truth_count = 579 class_id = 0, name = human_face, ap = 25.22% (TP = 152, FP = 212) for conf_thresh = 0.25, precision = 0.42, recall = 0.26, F1-score = 0.32 for conf_thresh = 0.25, TP = 152, FP = 212, FN = 427, average IoU = 27.22 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.252248, or 25.22 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
2000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_2000.weights... seen 64, trained: 128 K-images (2 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 7791, unique_truth_count = 579 class_id = 0, name = human_face, ap = 74.97% (TP = 378, FP = 70) for conf_thresh = 0.25, precision = 0.84, recall = 0.65, F1-score = 0.74 for conf_thresh = 0.25, TP = 378, FP = 70, FN = 201, average IoU = 59.01 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.749678, or 74.97 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
3000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_3000.weights... seen 64, trained: 192 K-images (3 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 7672, unique_truth_count = 579 class_id = 0, name = human_face, ap = 80.43% (TP = 447, FP = 102) for conf_thresh = 0.25, precision = 0.81, recall = 0.77, F1-score = 0.79 for conf_thresh = 0.25, TP = 447, FP = 102, FN = 132, average IoU = 60.25 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.804258, or 80.43 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

4000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_4000.weights... seen 64, trained: 256 K-images (4 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 3910, unique_truth_count = 579 class_id = 0, name = human_face, ap = 80.06% (TP = 429, FP = 63) for conf_thresh = 0.25, precision = 0.87, recall = 0.74, F1-score = 0.80 for conf_thresh = 0.25, TP = 429, FP = 63, FN = 150, average IoU = 66.00 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.800639, or 80.06 % Total Detection Time: 21 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
5000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_5000.weights... seen 64, trained: 320 K-images (5 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 4627, unique_truth_count = 579 class_id = 0, name = human_face, ap = 81.88% (TP = 451, FP = 87) for conf_thresh = 0.25, precision = 0.84, recall = 0.78, F1-score = 0.81 for conf_thresh = 0.25, TP = 451, FP = 87, FN = 128, average IoU = 65.34 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.818787, or 81.88 % Total Detection Time: 21 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
6000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_6000.weights... seen 64, trained: 384 K-images (6 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 4435, unique_truth_count = 579 class_id = 0, name = human_face, ap = 87.02% (TP = 476, FP = 92) for conf_thresh = 0.25, precision = 0.84, recall = 0.82, F1-score = 0.83 for conf_thresh = 0.25, TP = 476, FP = 92, FN = 103, average IoU = 65.62 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.870162, or 87.02 % Total Detection Time: 5 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

Table B-2: Output of YOLOv4 data training for Adience

Number of iterations	Output
1000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_1000.weights... seen 64, trained: 64 K-images (1 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 6989, unique_truth_count = 579 class_id = 0, name = human_face, ap = 30.12% (TP = 148, FP = 36) for conf_thresh = 0.25, precision = 0.80, recall = 0.26, F1-score = 0.39 for conf_thresh = 0.25, TP = 148, FP = 36, FN = 431, average IoU = 60.37 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.301174, or 30.12 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
2000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_2000.weights... seen 64, trained: 128 K-images (2 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 3268, unique_truth_count = 579 class_id = 0, name = human_face, ap = 39.25% (TP = 214, FP = 45) for conf_thresh = 0.25, precision = 0.83, recall = 0.37, F1-score = 0.51 for conf_thresh = 0.25, TP = 214, FP = 45, FN = 365, average IoU = 64.30 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.392513, or 39.25 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
3000	<pre> [yolov4] params: 100 loss: C100 (#), 100 norm: 0.07, obj_norm: 1.00, cls_norm: nms_kind: greedy (1), beta = 0.600000 Total BFLOPS 59.364 avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_3000.weights... seen 64, trained: 192 K-images (3 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 3761, unique_truth_count = 579 class_id = 0, name = human_face, ap = 41.82% (TP = 231, FP = 90) for conf_thresh = 0.25, precision = 0.72, recall = 0.40, F1-score = 0.51 for conf_thresh = 0.25, TP = 231, FP = 90, FN = 348, average IoU = 58.97 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.418161, or 41.82 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

4000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_4000.weights... seen 64, trained: 256 K-images (4 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 1545, unique_truth_count = 579 class_id = 0, name = human_face, ap = 43.05% (TP = 221, FP = 19) for conf_thresh = 0.25, precision = 0.92, recall = 0.38, F1-score = 0.54 for conf_thresh = 0.25, TP = 221, FP = 19, FN = 358, average IoU = 74.69 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.430452, or 43.05 % Total Detection Time: 21 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
5000	<pre> [yolo] params: iou_loss: ciou (4), iou_norm: 0.0/, obj_norm: 1.00, nms_kind: greedy (1), beta = 0.600000 Total BFLOPS 59.364 avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_5000.weights... seen 64, trained: 320 K-images (5 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 1414, unique_truth_count = 579 class_id = 0, name = human_face, ap = 44.98% (TP = 234, FP = 17) for conf_thresh = 0.25, precision = 0.93, recall = 0.40, F1-score = 0.56 for conf_thresh = 0.25, TP = 234, FP = 17, FN = 345, average IoU = 80.37 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.449758, or 44.98 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
6000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_final.weights... seen 64, trained: 384 K-images (6 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 1394, unique_truth_count = 579 class_id = 0, name = human_face, ap = 45.91% (TP = 230, FP = 24) for conf_thresh = 0.25, precision = 0.91, recall = 0.40, F1-score = 0.55 for conf_thresh = 0.25, TP = 230, FP = 24, FN = 349, average IoU = 78.90 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.459070, or 45.91 % Total Detection Time: 5 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

Table B-3: Output of YOLOv4 data training
for Labeled Faces in the Wild (LFW)

Number of iterations	Output
1000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_1000.weights... seen 64, trained: 64 K-images (1 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 12525, unique_truth_count = 579 class_id = 0, name = human_face, ap = 22.38% (TP = 162, FP = 299) for conf_thresh = 0.25, precision = 0.35, recall = 0.28, F1-score = 0.31 for conf_thresh = 0.25, TP = 162, FP = 299, FN = 417, average IoU = 23.74 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.223831, or 22.38 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
2000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_2000.weights... seen 64, trained: 128 K-images (2 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 5814, unique_truth_count = 579 class_id = 0, name = human_face, ap = 48.18% (TP = 254, FP = 83) for conf_thresh = 0.25, precision = 0.75, recall = 0.44, F1-score = 0.55 for conf_thresh = 0.25, TP = 254, FP = 83, FN = 325, average IoU = 58.12 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.481779, or 48.18 % Total Detection Time: 21 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
3000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_3000.weights... seen 64, trained: 192 K-images (3 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 2403, unique_truth_count = 579 class_id = 0, name = human_face, ap = 48.91% (TP = 256, FP = 40) for conf_thresh = 0.25, precision = 0.86, recall = 0.44, F1-score = 0.59 for conf_thresh = 0.25, TP = 256, FP = 40, FN = 323, average IoU = 71.52 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.489064, or 48.91 % Total Detection Time: 21 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

4000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_4000.weights... seen 64, trained: 256 K-images (4 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 4281, unique_truth_count = 579 class_id = 0, name = human_face, ap = 56.58% (TP = 290, FP = 56) for conf_thresh = 0.25, precision = 0.84, recall = 0.50, F1-score = 0.63 for conf_thresh = 0.25, TP = 290, FP = 56, FN = 289, average IoU = 67.19 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.565789, or 56.58 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
5000	<pre> Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_5000.weights... seen 64, trained: 320 K-images (5 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 1767, unique_truth_count = 579 class_id = 0, name = human_face, ap = 53.32% (TP = 260, FP = 16) for conf_thresh = 0.25, precision = 0.94, recall = 0.45, F1-score = 0.61 for conf_thresh = 0.25, TP = 260, FP = 16, FN = 319, average IoU = 79.82 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.533206, or 53.32 % Total Detection Time: 22 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
6000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_6000.weights... seen 64, trained: 384 K-images (6 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 1377, unique_truth_count = 579 class_id = 0, name = human_face, ap = 51.21% (TP = 254, FP = 19) for conf_thresh = 0.25, precision = 0.93, recall = 0.44, F1-score = 0.60 for conf_thresh = 0.25, TP = 254, FP = 19, FN = 325, average IoU = 79.81 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.512114, or 51.21 % Total Detection Time: 5 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

Table B-4: Output of YOLOv4 data training
for Unconstrained Facial Images (UFI)

Number of iterations	Output
1000	<pre> [yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: nms_kind: greedy (1), beta = 0.600000 Total BFLOPS 59.364 avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_1000.weights... seen 64, trained: 64 K-images (1 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 12051, unique_truth_count = 579 class_id = 0, name = human_face, ap = 11.96% (TP = 97, FP = 114) for conf_thresh = 0.25, precision = 0.46, recall = 0.17, F1-score = 0.25 for conf_thresh = 0.25, TP = 97, FP = 114, FN = 482, average IoU = 31.99 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.119552, or 11.96 % Total Detection Time: 24 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
2000	<pre> avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_2000.weights... seen 64, trained: 128 K-images (2 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 2349, unique_truth_count = 579 class_id = 0, name = human_face, ap = 21.05% (TP = 136, FP = 174) for conf_thresh = 0.25, precision = 0.44, recall = 0.23, F1-score = 0.31 for conf_thresh = 0.25, TP = 136, FP = 174, FN = 443, average IoU = 32.11 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.210452, or 21.05 % Total Detection Time: 23 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
3000	<pre> Total BFLOPS 59.364 avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_3000.weights... seen 64, trained: 192 K-images (3 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 2492, unique_truth_count = 579 class_id = 0, name = human_face, ap = 24.30% (TP = 148, FP = 124) for conf_thresh = 0.25, precision = 0.54, recall = 0.26, F1-score = 0.35 for conf_thresh = 0.25, TP = 148, FP = 124, FN = 431, average IoU = 41.30 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.242997, or 24.30 % Total Detection Time: 23 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

4000	<pre> Total BFLOPS 59.364 avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_4000.weights... seen 64, trained: 256 K-images (4 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 3395, unique_truth_count = 579 class_id = 0, name = human_face, ap = 25.01% (TP = 152, FP = 179) for conf_thresh = 0.25, precision = 0.46, recall = 0.26, F1-score = 0.33 for conf_thresh = 0.25, TP = 152, FP = 179, FN = 427, average IoU = 34.08 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.250095, or 25.01 % Total Detection Time: 23 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
5000	<pre> avg_outputs = 489778 Allocate additional workspace_size = 12.46 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_5000.weights... seen 64, trained: 320 K-images (5 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 1528, unique_truth_count = 579 class_id = 0, name = human_face, ap = 21.75% (TP = 143, FP = 176) for conf_thresh = 0.25, precision = 0.45, recall = 0.25, F1-score = 0.32 for conf_thresh = 0.25, TP = 143, FP = 176, FN = 436, average IoU = 34.06 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.217535, or 21.75 % Total Detection Time: 23 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
6000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_6000.weights... seen 64, trained: 384 K-images (6 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 4622, unique_truth_count = 579 class_id = 0, name = human_face, ap = 22.26% (TP = 138, FP = 232) for conf_thresh = 0.25, precision = 0.37, recall = 0.24, F1-score = 0.29 for conf_thresh = 0.25, TP = 138, FP = 232, FN = 441, average IoU = 27.49 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.222638, or 22.26 % Total Detection Time: 6 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

Table B-5: Output of YOLOv4 data training for
Unconstrained Face Detection Dataset (UFDD)

Number of iterations	Output
1000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_1000.weights... seen 64, trained: 64 K-images (1 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 30035, unique_truth_count = 579 class_id = 0, name = human_face, ap = 17.11% (TP = 115, FP = 200) for conf_thresh = 0.25, precision = 0.37, recall = 0.20, F1-score = 0.26 for conf_thresh = 0.25, TP = 115, FP = 200, FN = 464, average IoU = 23.74 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.171097, or 17.11 % Total Detection Time: 7 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
2000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_2000.weights... seen 64, trained: 128 K-images (2 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 19260, unique_truth_count = 579 class_id = 0, name = human_face, ap = 62.15% (TP = 384, FP = 392) for conf_thresh = 0.25, precision = 0.49, recall = 0.66, F1-score = 0.57 for conf_thresh = 0.25, TP = 384, FP = 392, FN = 195, average IoU = 34.61 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.621506, or 62.15 % Total Detection Time: 6 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
3000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_3000.weights... seen 64, trained: 192 K-images (3 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 10936, unique_truth_count = 579 class_id = 0, name = human_face, ap = 72.63% (TP = 409, FP = 288) for conf_thresh = 0.25, precision = 0.59, recall = 0.71, F1-score = 0.64 for conf_thresh = 0.25, TP = 409, FP = 288, FN = 170, average IoU = 42.78 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.726337, or 72.63 % Total Detection Time: 6 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

4000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_4000.weights... seen 64, trained: 256 K-images (4 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 4010, unique_truth_count = 579 class_id = 0, name = human_face, ap = 77.38% (TP = 398, FP = 85) for conf_thresh = 0.25, precision = 0.82, recall = 0.69, F1-score = 0.75 for conf_thresh = 0.25, TP = 398, FP = 85, FN = 181, average IoU = 60.75 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.773771, or 77.38 % Total Detection Time: 5 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
5000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_5000.weights... seen 64, trained: 320 K-images (5 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 4186, unique_truth_count = 579 class_id = 0, name = human_face, ap = 80.86% (TP = 432, FP = 155) for conf_thresh = 0.25, precision = 0.74, recall = 0.75, F1-score = 0.74 for conf_thresh = 0.25, TP = 432, FP = 155, FN = 147, average IoU = 56.95 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.808631, or 80.86 % Total Detection Time: 5 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>
6000	<pre> Allocate additional workspace_size = 52.43 MB Loading weights from /mydrive/yolov4/backup/yolov4-obj_6000.weights... seen 64, trained: 384 K-images (6 Kilo-batches_64) Done! Loaded 162 layers from weights-file calculation mAP (mean average precision)... Detection layer: 139 - type = 28 Detection layer: 150 - type = 28 Detection layer: 161 - type = 28 300 detections_count = 3804, unique_truth_count = 579 class_id = 0, name = human_face, ap = 81.63% (TP = 435, FP = 121) for conf_thresh = 0.25, precision = 0.78, recall = 0.75, F1-score = 0.77 for conf_thresh = 0.25, TP = 435, FP = 121, FN = 144, average IoU = 60.78 % IoU threshold = 50 %, used Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) = 0.816298, or 81.63 % Total Detection Time: 5 Seconds Set -points flag: `-points 101` for MS COCO `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data) `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset </pre>

Table B-6: Output of YOLOv5 data training for Open Images Dataset V6

Number of iterations	Output									
1000	Epoch 43/255	gpu_mem 5.75G	box 0.05052	obj 0.02618	cls 0	total 0.0767	targets 131	img_size 416:	100% 24/24 [00:09<00:00, 2.54it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.00it/s]	
	all		300	578		0.805	0.55	0.635	0.259	
2000	Epoch 86/255	gpu_mem 5.75G	box 0.04046	obj 0.02199	cls 0	total 0.06245	targets 149	img_size 416:	100% 24/24 [00:09<00:00, 2.50it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.93it/s]	
	all		300	578		0.799	0.751	0.814	0.351	
3000	Epoch 128/255	gpu_mem 5.75G	box 0.03452	obj 0.01885	cls 0	total 0.05337	targets 142	img_size 416:	100% 24/24 [00:09<00:00, 2.53it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.98it/s]	
	all		300	578		0.911	0.747	0.849	0.39	
4000	Epoch 171/255	gpu_mem 5.75G	box 0.03197	obj 0.01797	cls 0	total 0.04994	targets 131	img_size 416:	100% 24/24 [00:09<00:00, 2.55it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.98it/s]	
	all		300	578		0.897	0.753	0.841	0.398	
5000	Epoch 214/255	gpu_mem 5.75G	box 0.02913	obj 0.01674	cls 0	total 0.04587	targets 129	img_size 416:	100% 24/24 [00:09<00:00, 2.52it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.95it/s]	
	all		300	578		0.92	0.763	0.861	0.414	
6000	Epoch 255/255	gpu_mem 5.75G	box 0.02894	obj 0.01618	cls 0	total 0.04512	targets 138	img_size 416:	100% 24/24 [00:09<00:00, 2.55it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:02<00:00, 1.01it/s]	
	all		300	578		0.87	0.777	0.841	0.403	
	Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.7MB									
	Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.7MB									
	256 epochs completed in 0.829 hours.									
	CPU times: user 31 s, sys: 3.77 s, total: 34.8 s									
	Wall time: 50min 18s									

Table B-7: Output of YOLOv5 data training for Adience

Number of iterations	Output									
1000	Epoch 43/255	gpu_mem 5.76G	box 0.03358	obj 0.01589	cls 0	total 0.04947	targets 39	img_size 416:	100% 23/23 [00:09<00:00, 2.54it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.02it/s]	
	all		300	578		0.616	0.285	0.305	0.137	
2000	Epoch 86/255	gpu_mem 5.76G	box 0.02442	obj 0.01219	cls 0	total 0.03661	targets 45	img_size 416:	100% 23/23 [00:09<00:00, 2.49it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.04it/s]	
	all		300	578		0.803	0.472	0.539	0.302	
3000	Epoch 128/255	gpu_mem 5.76G	box 0.02011	obj 0.0106	cls 0	total 0.03071	targets 36	img_size 416:	100% 23/23 [00:08<00:00, 2.57it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.99it/s]	
	all		300	578		0.792	0.481	0.544	0.313	
4000	Epoch 171/255	gpu_mem 5.76G	box 0.01871	obj 0.0101	cls 0	total 0.02881	targets 49	img_size 416:	100% 23/23 [00:09<00:00, 2.55it/s]	
	Class		Images	Targets		P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.07it/s]	
	all		300	578		0.878	0.512	0.601	0.364	

5000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	214/255	5.76G	0.01614	0.009373	0	0.02551	43	416: 100% 23/23 [00:09<00:00, 2.55it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.97it/s]	
	all	300	578	0.828	0.552	0.613	0.365	

6000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	255/255	5.76G	0.01594	0.008821	0	0.02476	39	416: 100% 23/23 [00:08<00:00, 2.58it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:02<00:00, 1.05it/s]	
	all	300	578	0.872	0.509	0.608	0.344	
Optimizer stripped from runs/train/yolov5s_results2/weights/last.pt, 14.7MB								
Optimizer stripped from runs/train/yolov5s_results2/weights/best.pt, 14.7MB								
256 epochs completed in 0.789 hours.								
CPU times: user 30.8 s, sys: 3.63 s, total: 34.5 s								
Wall time: 47min 42s								

Table B-8: Output of YOLOv5 data training
for Labeled Faces in the Wild (LFW)

Number of iterations	Output							
1000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	43/255	5.74G	0.03199	0.01402	0	0.04601	67	416: 100% 24/24 [00:09<00:00, 2.49it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.87it/s]	
	all	300	578	0.657	0.335	0.36	0.177	
2000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	86/255	5.74G	0.02361	0.01156	0	0.03517	62	416: 100% 24/24 [00:09<00:00, 2.50it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.02it/s]	
	all	300	578	0.847	0.452	0.518	0.296	
3000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	128/255	5.74G	0.0204	0.009969	0	0.03037	76	416: 100% 24/24 [00:09<00:00, 2.51it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.05it/s]	
	all	300	578	0.855	0.46	0.541	0.32	
4000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	171/255	5.74G	0.018	0.009602	0	0.02761	66	416: 100% 24/24 [00:09<00:00, 2.49it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.93it/s]	
	all	300	578	0.898	0.488	0.587	0.339	
5000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	214/255	5.74G	0.01672	0.008965	0	0.02569	71	416: 100% 24/24 [00:09<00:00, 2.50it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.02it/s]	
	all	300	578	0.819	0.557	0.629	0.342	
6000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	255/255	5.74G	0.01586	0.008684	0	0.02455	73	416: 100% 24/24 [00:09<00:00, 2.51it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:02<00:00, 1.06it/s]	
	all	300	578	0.808	0.54	0.613	0.339	
Optimizer stripped from runs/train/yolov5s_results2/weights/last.pt, 14.7MB								
Optimizer stripped from runs/train/yolov5s_results2/weights/best.pt, 14.7MB								
256 epochs completed in 0.833 hours.								
CPU times: user 26.6 s, sys: 3.29 s, total: 29.9 s								
Wall time: 50min 19s								

Table B-9: Output of YOLOv5 data training
for Unconstrained Facial Images (UFI)

Number of iterations	Output							
1000	Epoch	gpu_mem	box	obj	cls	total	targets	img_size
	43/255	5.74G	0.02187	0.01096	0	0.03283	67	416: 100% 24/24 [00:09<00:00, 2.52it/s]
	Class	Images	Targets	P	R	mAP@.5	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.05it/s]	
	all	300	578	0.382	0.195	0.137	0.0583	

2000	Epoch 86/255	gpu_mem 5.74G	box 0.01465	obj 0.008358	cls 0	total 0.02301	targets 63	img_size 416: 100% 24/24 [00:09<00:00, 2.52it/s]
	Class all	Images 300	Targets 578	P 0.545	R 0.239	mAP@.5 0.22	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.00it/s]	0.112
3000	Epoch 128/255	gpu_mem 5.74G	box 0.0127	obj 0.007243	cls 0	total 0.01994	targets 64	416: 100% 24/24 [00:09<00:00, 2.51it/s]
	Class all	Images 300	Targets 578	P 0.556	R 0.242	mAP@.5 0.225	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.89it/s]	0.125
4000	Epoch 171/255	gpu_mem 5.74G	box 0.01142	obj 0.006518	cls 0	total 0.01793	targets 71	416: 100% 24/24 [00:09<00:00, 2.48it/s]
	Class all	Images 300	Targets 578	P 0.555	R 0.268	mAP@.5 0.242	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.91it/s]	0.132
5000	Epoch 214/255	gpu_mem 5.74G	box 0.009873	obj 0.006117	cls 0	total 0.01599	targets 66	416: 100% 24/24 [00:09<00:00, 2.50it/s]
	Class all	Images 300	Targets 578	P 0.7	R 0.299	mAP@.5 0.294	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.02it/s]	0.15
6000	Epoch 255/255	gpu_mem 5.74G	box 0.008867	obj 0.005933	cls 0	total 0.0148	targets 67	416: 100% 24/24 [00:09<00:00, 2.51it/s]
	Class all	Images 300	Targets 578	P 0.684	R 0.289	mAP@.5 0.281	mAP@.5:.95: 100% 3/3 [00:02<00:00, 1.04it/s]	0.149
	Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.7MB Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.7MB 256 epochs completed in 0.828 hours.							
	CPU times: user 32 s, sys: 3.73 s, total: 35.7 s Wall time: 50min 12s							

Table B-10: Output of YOLOv5 data training for
Unconstrained Face Detection Dataset (UFDD)

Number of iterations	Output							
1000	Epoch 43/255	gpu_mem 5.4G	box 0.04479	obj 0.01726	cls 0	total 0.06205	targets 192	img_size 416: 100% 47/47 [00:19<00:00, 2.43it/s]
	Class all	Images 300	Targets 578	P 0.892	R 0.602	mAP@.5 0.712	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.71it/s]	0.325
2000	Epoch 86/255	gpu_mem 5.4G	box 0.03634	obj 0.01444	cls 0	total 0.05077	targets 173	416: 100% 47/47 [00:19<00:00, 2.44it/s]
	Class all	Images 300	Targets 578	P 0.879	R 0.683	mAP@.5 0.792	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.60it/s]	0.402
3000	Epoch 128/255	gpu_mem 5.4G	box 0.03143	obj 0.01272	cls 0	total 0.04416	targets 200	416: 100% 47/47 [00:19<00:00, 2.43it/s]
	Class all	Images 300	Targets 578	P 0.904	R 0.73	mAP@.5 0.853	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.91it/s]	0.458
4000	Epoch 171/255	gpu_mem 5.4G	box 0.02778	obj 0.01167	cls 0	total 0.03945	targets 217	416: 100% 47/47 [00:19<00:00, 2.41it/s]
	Class all	Images 300	Targets 578	P 0.864	R 0.772	mAP@.5 0.849	mAP@.5:.95: 100% 3/3 [00:01<00:00, 1.80it/s]	0.477
5000	Epoch 214/255	gpu_mem 5.4G	box 0.02504	obj 0.01056	cls 0	total 0.0356	targets 211	416: 100% 47/47 [00:19<00:00, 2.47it/s]
	Class all	Images 300	Targets 578	P 0.887	R 0.749	mAP@.5 0.841	mAP@.5:.95: 100% 3/3 [00:01<00:00, 2.01it/s]	0.474

6000

```

Epoch  gpu_mem  box  obj  cls  total  targets  img_size
255/255  5.4G  0.02338  0.01015  0  0.03353  239  416: 100% 47/47 [00:19<00:00, 2.46it/s]
      Class  Images  Targets  P  R  mAP@.5  mAP@.5:.95: 100% 3/3 [00:02<00:00, 1.04it/s]
      all    300    578    0.874  0.782  0.854  0.484
Optimizer stripped from runs/train/yolov5s_results2/weights/last.pt, 14.7MB
Optimizer stripped from runs/train/yolov5s_results2/weights/best.pt, 14.7MB
256 epochs completed in 1.547 hours.

CPU times: user 53.6 s, sys: 6.48 s, total: 1min
Wall time: 1h 33min 13s

```