

A COMPLETE DEEP LEARNING-BASED DRIVER MONITORING SYSTEM WITH LOW COST

by

BING LIU

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2026

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Subramaniam Ganesan, Ph.D., Chair
Xiaodong Deng, Ph.D.
Anyi Liu, Ph.D.
Shadi Alawneh, Ph.D.

© Copyright By Bing Liu, 2026
All rights reserved

ACKNOWLEDGMENTS

This dissertation proposal and research finished has been fully supported by my committee members.

First, I would like to thank my advisor, Dr. Subramaniam Ganesan, for his rich knowledge, high standard of academic requirements, candid advice, and support of my research work.

I would like to acknowledge my dissertation committee members, Dr. Xiaodong Deng, Dr. Anyi Liu and Dr. Shadi Alawneh, for their guidance during my study at Oakland University.

Bing Liu

ABSTRACT

A COMPLETE DEEP LEARNING-BASED DRIVER MONITORING SYSTEM WITH LOW COST

by

BING LIU

Adviser: Subramaniam Ganesan, Ph.D.

This research provides a solution to build a deep learning-based driving monitoring system (DDMS) to verify the driver's ID, estimate the driver's status and the vehicle's occupancy status.

In our DDMS, we built and trained three deep learning neural networks: object detection neural network (ODNN), facial recognition neural network (FRNN) and facial landmark detection neural network (FLDNN). We built software to drive a camera to capture images in the vehicle and integrate these three neural networks to build our DDMS. The pictures of the driver seat, passenger seat and back seat are sent to ODNN. ODNN detects common objects in vehicles such as humans, faces, dogs, and so on; the driver's facial images are sent to FRNN. Our DDMS uses the facial recognition result to verify if the person is authorized to drive; the driver's face images are also sent to FLDNN. FLDNN detects facial keypoints including eyebrows, eyes, the nose, lips, and the chin continuously. Our DDMS estimates driver's state such as activeness, drowsiness, distractedness and so on. The movement of lips detected by FLDNN and other detected

objects by ODNN near the driver's face such as cell phone are used to estimate if the driver is speaking on the phone.

Public datasets that meet our requirements to train our ODNN do not exist to the best of our knowledge. We built an Auto-Labeling Model (ALM) to expand an existing dataset by assigning labels to the data automatically.

Quantizing deep learning models from floating point to 8-bit or lower-bit scaled integer data types is required to reduce the cost of using DDMS. Quantization is applied to convert the parameters in ODNN, FRNN and FLDNN from floating point numbers to integer numbers.

The contribution in our study includes building and training an ALM to create a new dataset for ODNN; building and training ODNN, FRNN and FLDNN; developing software to drive the camera and integrating ODNN, FRNN and FLDNN together to provide vehicle's occupancy status, face ID and driver status estimation results; quantizing ODNN, FRNN and FLDNN.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	iii
ABSTRACT	iv
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xii
CHAPTER ONE	1
INTRODUCTION	1
1.1 Problem Statement	1
1.2 Dissertation Objective	3
1.3 Dissertation Outline	8
CHAPTER TWO	9
RELATED WORKS	9
2.1 Driver Monitoring	9
2.2 Object Detection	9
2.3 YOLO Deep Learning Neural Networks	11
2.4 COCO Dataset	11
2.5 WIDER FACE Dataset	12
2.6 Dataset Annotation	12
2.7 Facial Recognition	13

2.8 Facial Landmark Detection	15
2.9 Neural Network Quantization	16
CHAPTER THREE	17
OBJECT DETECTION AND AUTO LABELING	17
3.1 Introduction	17
3.2 Expand COCO Dataset Automatically	19
3.2.1 Auto-Labeling Model	20
3.2.2 Label Automatically	26
3.2.3 Labelling Results	27
3.3 COCO Dataset Expansion and Object Detection	28
3.4 Summary	31
CHAPTER FOUR	33
FACIAL RECOGNITION	33
4.1 Introduction	33
4.2 Technical Approach	33
4.2.1 FRNN Structure	37
4.2.2 Facial Recognition Dataset	39
4.2.3 FRNN Training	40
4.3 Experimental Results	43

4.3.1 FRNN Performance	43
4.4 Summary	45
CHAPTER FIVE	46
FACIAL LANDMARK DETECTION	46
5.1 Introduction	46
5.2 Technical Approach	46
5.2.1 FRNN Structure	46
5.2.2 Facial Landmark Detection Dataset	47
5.2.3 FDLNN Training	48
5.3 Experimental Results	49
5.3.1 FDLNN Performance	51
5.4 Summary	51
CHAPTER SIX	52
NEURAL NETWORK QUANTIZATION	52
6.1 Introduction	52
6.2 Technical Approach	53
6.3 Experimental Results	53
6.4 Summary	55
CHAPTER SEVEN	56

SUMMARY	56
7.1 Overall Performance	56
7.2 Conclusion	59
7.3 Future Work	60
PUBLISHED PAPER DURING THIS RESEARCH: Received the best paper award	61
REFERENCES	62

LIST OF FIGURES

Figure 1. DDSM system.	3
Figure 2. DDSM system software design.	5
Figure 3. Auto-labeling system.	20
Figure 4. The structure of ALM.	21
Figure 5. Training samples of WIDER FACE dataset.	23
Figure 6. ALM training loss.	24
Figure 7. ALM performance	25
Figure 8. ALM validation examples	26
Figure 9. Examples of faces detected by ALM in COCO 2017.	27
Figure 10. Examples of wrongly detected as faces by ALM in COCO 2017.	28
Figure 11. Distribution of expanded object detection dataset	29
Figure 12. Detection results of the expanded dataset.	30
Figure 13. Object detection performance on the expanded dataset.	31
Figure 14. Residual neural networks.	34
Figure 15. Standard convolution.	35
Figure 16 Depthwise separable convolution.	37
Figure 17 FRNN architecture design.	38
Figure 18 VGGFace2 example images.	40
Figure 19. Training environment.	42
Figure 20. Accuracy increases as training time elapses.	43
Figure 21. ROC of FRNN.	44

Figure 22. Facial recognition results.	45
Figure 23 Examples of WFLW dataset.	48
Figure 24 FLDNN training progress.	49
Figure 25 FDLNN detection results of validation dataset.	50
Figure 26. Facial landmark detection result.	50
Figure 27. Facial landmarks detect results before and after model quantization.	54
Figure 28. Output from DDMS.	56
Figure 29. The driver is texting.	58

LIST OF ABBREVIATIONS

ADAS	Advanced driver-assistance system
ALM	Auto-labeling model
CNN	Convolution neural network
COCO	Microsoft common objects in context
DMS	Driving monitoring system
DDMS	Deep learning-based driving monitoring system
FLDNN	Facial landmark detection neural network
FPGAs	Field programmable gate arrays
FRNN	Facial recognition neural network
HOG	Histogram of orientation gradient
LiDAR	Light detection and ranging
mAP	Mean average precision
ODNN	Object detection neural network
YOLO	You only look once
RADAR	Radio detection and ranging
SGD	Stochastic gradient descent algorithm
VOC	Visual object classes
WIDER	Web image dataset for event recognition

CHAPTER ONE

INTRODUCTION

1.1 Problem Statement

Vehicle accidents are often caused by human errors, such as driving with fatigue, distracted driving, driving under the influence, leaving children or pets in the vehicle without attention for long time, etc. To reduce such accidents, vehicle manufacturers and researchers developed various driver monitoring systems (DMS) that can evaluate the driver and provide warning or intervene if necessary. Most accidents could be avoided with the help of DMS.

DMS is designed to assess the state of the driver and keep everyone safe built with hardware and software. The hardware usually consists of a camera mounted on the dashboard facing the driver and a computer capable of image processing. The software in a DMS usually provides following features: 1) recognizing if a person is authorized to drive the vehicle; 2) estimate if the driver is distracted from driving; 3) detect the vehicle occupancy status if human subjects are present or not in the vehicle.

The first instance of a commercial driver monitoring system was made by Toyota for Lexus in 2006 [1]. It utilizes a driver-facing camera mounted on the steering column to determine if the driver is facing the road. According to Toyota, the number one cause of vehicle crashes is caused by drivers not focusing on the road. In recent years, General Motors, BMW, and other vehicle manufacturers also adopted their own driver monitoring systems. General Motors integrated Seeing Machines' FOVIO Driver Monitoring System into the 2017 Cadillac CT6 [2]. It enabled hands-free driving experience on highways. It

utilizes driver-facing infrared cameras to track the driver's facial features to evaluate their physical and mental state. BMW offered the Extended Traffic Jam Assistant System as an upgrade for the 2019 X5 [3]. Like General Motors, it is implemented as part of limited self-driving experience.

Besides vehicle manufacturers, researchers are also developing different driver monitoring systems. In the researchers of [4] proposed the usage of seatbelt vibration as a warning system and measuring the driver's eyelids' closeness to determine driver fatigue. They cited that the US Federal Highway Administration and US National Highway Traffic Safety Administration consider eye closure to be one of the most promising physical measures to measure driver fatigue. The system can provide warning in half of a second, which might not be enough to avoid an emergency. In this research work [5], the authors took it a step further and utilized facial landmark detection instead of limiting to eye measurements. They believe that facial features can be used to estimate a driver's heart rate which in turn can be used to measure the driver's state. They used independent vector analysis to mark facial features and tested the final product with five short videos. While the estimation and actual measurements are close, all actual heart rates are between 71 to 75 beats per minute. The performance of this system beyond these conditions are unknown.

While these commercial systems are different from each other, they are all limited to monitoring the state of the driver and ignoring other parts of the car's interior. The system introduced in this research will build upon prior work to increase performance and implement new features.

Since the field of driver monitoring [6] has many important applications, ranging from identifying unauthorized person driving the vehicle, leaving babies or pets alone in the vehicle, to incident prevention on the road caused by distractive driver. So, a complete solution of DMS is desired.

1.2 Dissertation Objective

The objective of this dissertation focuses on providing a complete solution to build a deep learning-based driving monitoring system (DDMS). All the features of DMS mentioned above are studied in our work. This system is depicted in Figure 1.

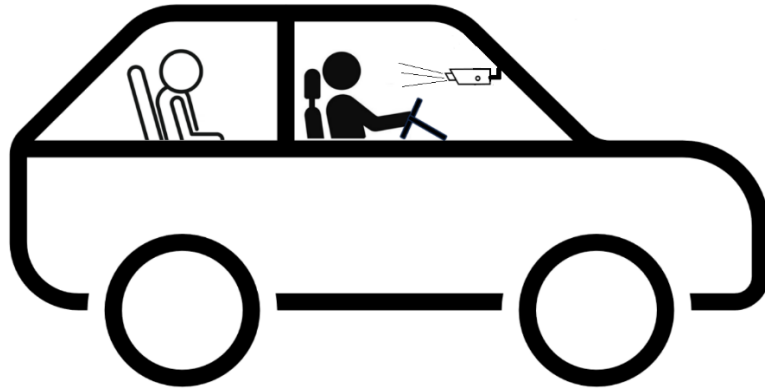


Figure 1. DDSM system.

First, a camera is installed in front of driver seat and takes the pictures inside of the vehicle continuously. Second, the software and three deep learning neural network models are built. The software drives the DDMS and makes use of these three deep learning neural network models. These models are responsible for object detection, facial recognition, and facial landmark detection. The responsibility of the software and each

neural network model is: 1) the software drives the camera to actively to take pictures of the driver seat, passenger seat and back seat; 2) the captured images are sent to the object detection neural network (ODNN). ODNN detects the human subjects, faces, dogs, and other common objects in the vehicle. 3) the detected face images by ODNN on the driver seat are sent to the facial recognition neural network (FRNN), then the software uses the recognition result from FRNN to verify if the person is authorized to drive; 4) the driver's face images are sent to facial landmark detection neural network (FLDNN), then FLDNN detects the facial keypoints including eyebrows, the eyes, the nose, the lips, and the chin continuously; 5) the software estimates the state of the driver such activeness, drowsiness, distractedness and so on using detected facial keypoints and object detection results.

The software system design is displayed in Figure 2. The software drives the camera to actively take pictures and integrate these three deep learning models together to provide vehicle occupancy status, face ID identification results and driver status estimation results. The software takes pictures of the driver's seat, passenger seat and back seat. The captured images are preprocessed then are sent to ODNN. ODNN provides object detection results such as the present status of human subjects, faces, dogs, and other common objects in the vehicle. The object detection model in the system outputs the vehicle occupancy status based on the object detection results such as if the vehicle is empty, occupied by human or pets and so on.

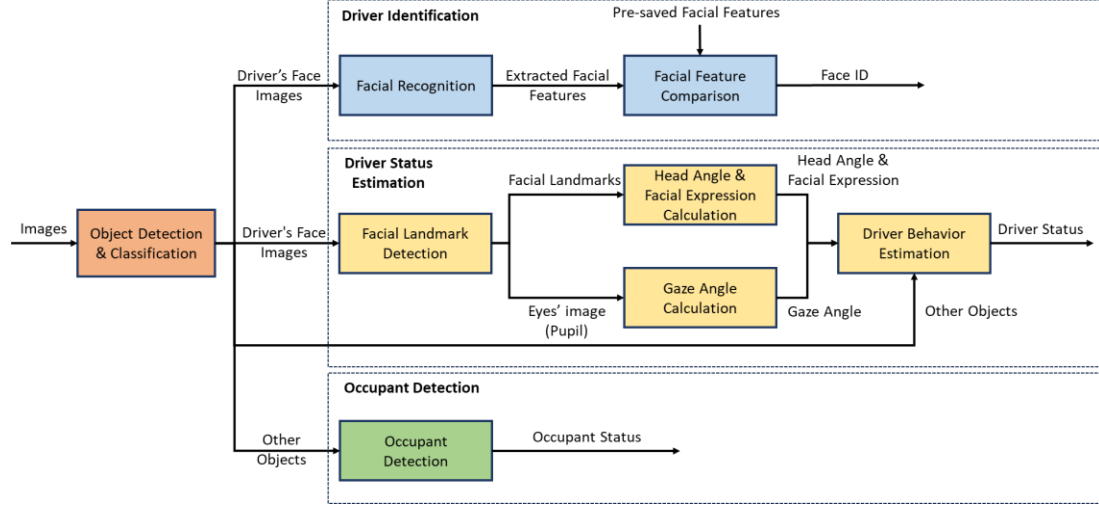


Figure 2. DDSM system software design.

ODNN also provides the positions of the detected objects on the images. The positions of detected objects relative to the vehicle can be calculated since the camera and focus point are fixed. So, the driver's face can be located based on the relative position of objects detected on the image. The software extracts the driver's facial image from the picture when a person and a face are detected at the driver seat by ODNN. The extracted facial image is sent to FRNN and FLDNN.

FRNN takes the driver's facial image as the input and goes through the feed forward process to calculate the features of the driver's face. The software in the system calculates the distance between the driver's face features obtained by FRNN and the pre-saved feature of each driver's face. Thus, the driver's face ID can be identified when distance is lower than the threshold.

FLDNN takes the driver's facial image as the input and goes through the feed forward process to detect the facial landmarks. These facial landmarks are used by the

software to detect the driver's status. The head angle and facial expression can be calculated based on the relative position of the detected eyebrows, the eyes, the nose, the lips, and the chin. Then, the driver status can be estimated accordingly. For example, the software can conclude that the driver is talking when the lips open and close regularly. Other status such as sleepiness can be detected based on relative position changes using the points of eyes. Through image process, the pupil can be located so the gaze angle can be calculated. The status such as if the driver is looking around or focusing on driving and so on can be detected.

The software also takes objects detected by ODNN as the input to detect driver status. For example, the software can conclude that the driver is looking at the phone when a cell phone is detected near the driver's face and gaze angle is towards the cell phone. Other status such as eating can be detected as well when a sandwich is near the driver's face and the lips close and open regularly. A complete DDMS solution and investigation is proposed in this dissertation to detect driver status of activeness, drowsiness, distractedness and so on.

The high performance such as accuracy of the trained models is the key to building an efficient DDMS. All these three models need to be trained with large amounts of data. Enough and high-quality datasets are available to train FRNN and FLDNN. However, to train ODNN in this system, we need labelled objects on the images including human faces, persons, mobile phones, pets, foods and so on. Such publicly available datasets which can meet our requirements do not exist to the best of our knowledge when this research is being conducted. It is expensive to create or expand an existing dataset manually to satisfy this research goal. So, an Auto-Labeling Model

(ALM) is built to expand an existing dataset by assigning labels to the data automatically. A new dataset including all objects needed in this study is created and is used to train ODNN. Human faces along other objects such as person, mobile phones, pets, foods and so on are detected automatically.

The software and trained models need to be executed in the embedded system in the vehicle. It is desired to run a DDMS using low-end hardware to reduce the cost but keep high running speed and high accuracy. These three models are trained using floating points numbers so the parameters and the output from activation functions are floating points numbers. Running these neural networks requires certain amount of memory space and needs high-end hardware that must support floating-point arithmetic to maintain the high running speed. So, deploying DDMS to low-power microcontrollers and Field Programmable Gate Arrays (FPGAs) is prohibited. Quantizing deep learning models from floating point to 8-bit or lower-bit scaled integer data types is required in the industrial world to reduce the cost. Quantization is applied in this research to convert the parameters in these models from floating point numbers to integer numbers without losing much accuracy. The size of the model is reduced to one fourth and the execution speed is nearly doubled after quantization.

The scope of this study includes: 1) train an ALM to create a new dataset which is used to train ODNN; 2) train three neural networks including ODNN, FRNN and FLDNN which are used to detect common objects in the vehicle, identify face ID and detect facial landmarks of the driver face; 3) build a software to drive the camera and integrate ODNN, FRNN and FLDNN together by using the images captured to provide vehicle occupancy status, face ID and driver status estimation results; 4) Quantize

ODNN, FRNN and FLDNN are provided which can be used in a low cost execution environment.

1.3 Dissertation Outline

The rest of this dissertation is organized as follows. Chapter Two introduces the related works. Chapter Three presents the object detection model ODNN and auto-labeling model ALM. Chapter Four and Five detail the facial recognition and facial landmark detection. Chapter Six depicts the model quantization. Finally, Chapter Seven concludes the study and points out future research directions.

CHAPTER TWO

RELATED WORKS

2.1 Driver Monitoring

Driver monitoring is part of an advanced driving assistance system (ADAS). In a DMS, a driver-facing camera is usually integrated into the dashboard or instrument cluster. Sometimes, it is also equipped with infrared light-emitting diodes (LEDs) to take pictures of the driver's face in the dark environment. DMS collects data of the facial features when the driver is awake and attentive. These collected data are used to create an initial baseline. Then, the base line is used to compare with real time facial features. DMS determines if the driver is blinking more, if the eyes are narrowing or closing, or if the head is leaning to an unnormal angle based on the comparison results. DMS can also estimate if the driver looks at the road ahead or is distracted from driving. DMS alters the driver if such situations appear. Deep learning-based approaches are widely used in DMS due to its outstanding performance. Other features such as driver authentic verification and preventing kids or pets from being left in the vehicle alone are also preferred.

2.2 Object Detection

In computer vision, object detection is an important task that has been studied intensively in the recent two decades. The goal is to develop algorithms and build models to detect objects in the images, estimate the classes and outline the objects using bounding boxes [7]. Comparing to other object detection methods utilizing sensors, such as Light Detection and Ranging (LiDAR) [8] or Radio Detection and Ranging (RADAR)

[9], vision-based objects detection method using camera has the advantages of being low-cost, flexible, easy to deploy, and does not produce wireless signals pollution to our already crowded Radio Frequency (RF) space. Object detection via camera has been studied by many scholars and shown its power in many applications despite heavily relying on the visible spectrum. The applications, such as object tracking [10] and moving object detection [11], in the domain of video surveillance can protect our lives and properties from dangerous. It can also be applied to inventory stock counting [12], high performance video analysis [13], counter-less checking out for smart retail [14], detecting and classifying animals on hunting cameras [15], recognizing objects in hospitals and health care centers without violating patient privacy [16], and autonomous driving with sensor fusion [17].

Due to the wide applications and importance of object detection, different algorithms are developed to address this need. Histogram of Orientation Gradient (HOG) was proposed by Dalal and Triggs in 2005 [18]. This sophisticated traditional method did not involve large amount of training data, but its maximum performance is limited due to its design. Similar research works conducted by other scholars all fall into this category [19], [20], [21], [22]. Along with the development of Convolutional Neural Networks (CNN), Girshick et al. applied CNN to object detection and a breakthrough was made in 2014. The best performance was achieved by measuring mean average precision (mAP) on Visual Object Classes (VOC) 2012 dataset in 2014. CNN is a supervised machine learning algorithm which needs larger amount of data but less handcrafting. Fast R-CNN was proposed by Girshick et al. in 2015 with both speed and performance improved. Both above mentioned neural networks fall into the classification of two-stage detectors.

2.3 YOLO Deep Learning Neural Networks

YOLO deep learning neural networks was proposed by Joseph et al. in 2015 which was the first one-stage detector in the deep learning. YOLO is extremely fast compared to other deep learning neural network-based algorithms. YOLO applies to a single neural network to the full image instead of using two-stage detectors. An image is divided into multiple regions with bounding boxes. Yolo predicts these bounding boxes and estimates the probabilities for each region simultaneously, thus achieves amazingly fast speed. Same as other deep learning neural network-based models, it requires substantial amounts of training data.

2.4 COCO Dataset

COCO is a large-scale image dataset initially published in 2014 which can be downloaded from <https://www.kaggle.com/datasets/awsaf49/coco-2017-dataset>. It collects images of common objects and humans from everyday lives to address the research goals of scene understanding, such as non-iconic views of objects detection, contextual reasoning, and precise 2D objects localization. It contains bounding boxes and segmentations. In this research, only bounding boxes are used for object detection. Images and corresponding annotations concerned by this study were released from 2014 to 2017. The latest version of 2017 is used in this research work. Human object is one of its 80 classes. But face is not an individual class.

2.5 WIDER FACE Dataset

WIDER FACE is an effective and richly annotated face detection training source dataset initially published by Chinese University of Hong Kong in 2015 which can be downloaded from <http://shuoyang1213.me/WIDERFACE/>. Faces are annotated by bounding boxes. Image in this dataset varies in scale, pose, and occlusion, including 61 event classes, such as festival, parade, demonstration, ceremony, marching, and so on. It has a broad range of faces from height less than 20 pixels to several hundred pixels.

2.6 Dataset Annotation

It is not possible to build an effective supervised machine learning model without a labeled dataset. The quality of dataset has significant impact on training results since machine learning algorithms learn from the data and corresponding labels. The data can be images, text, audio, video, and other formats. Data annotation is the process of assigning relevant tags to the data. It has several types, such as text annotation, sentiment annotation, bounding box, 3D cuboids, and so on. The job of data annotators is to analyze data based on human judgment and provide corresponding labels. Even though different commercial labeling systems are developed, human effort still is heavily involved due to the nature of the work. To train a deep learning neural network, usually millions of data are needed. It is necessary to build a tool which can automatically label the data with minimum human involvement.

2.7 Facial Recognition

Facial recognition is the process of identifying human faces within images or videos and matching these faces with their identity. It has many applications such as using facial recognition to identify individuals during system logging in to strengthen user authentication activity and other fields such as digital device encryption, driver monitoring and law enforcement and so on. Compared to other biometrics identification, such as fingerprints, iris, and voice, facial recognition is contactless and does not require the active participation of the subjects being identified. Facial recognition only requires a source of images or video feeds and connection to established facial databases.

The first instance of facial recognition was developed in the 1960s [23]. Their work consisted of manually marking various facial features, such as iris, lips, nose, and feeding their mathematical relations into computers to adjust for face angle, pose, etc. and identify the face. Due to the project's tie with intelligence agencies, the research was mostly unpublished. Takeo Kanade, one of the world's foremost researchers in computer vision, demonstrated an improved system of face recognition that can mark facial features automatically without human intervention [24]. This is a drastic improvement as the system is no longer limited by human efforts. However, the accuracy of facial feature marking is inferior compared to human manual marking.

In 2004, Qing, Shan, and Gao proposed the usage of a novel approach to face recognition to achieve consistent results under different lighting [25]. At the time of their publication, the recognition rate under outdoor lighting is low due to bright sunlight coming from different angles. To adjust for this, they calculated the possible result of each face under different lighting and to find the closest match. They achieved ideal

accuracy. However, all images tested are faces directly facing the camera. While most face recognition systems are based on black and white or RGB images, Li et al. proposed a system utilizing near-infrared images [26]. They assume the users are cooperative, meaning that will face the camera and ensure they are within the frame in the desired distance. Their hardware consists of near-infrared light-emitting diodes mounted on a camera that only allows near-infrared light to pass through. They also used the Adaptive Boosting machine learning algorithm to select the most effective feature for detection and construct classifier for face detection. The accuracy of this system was not discussed in their paper. In recent years, easy-to-use tools like Python and OpenCV, an image processing library, became widely spread in image processing, including face recognition. Sirivarshitha et al. were able to accomplish face normalization, feature extraction, and face matching using OpenCV and Python built in face detection library without extensive coding [27]. This demonstrated the improvement in tools available to researchers for face recognition and image processing in general.

Now, it is very popular on mobile phones to use facial recognition for device unlocking. Facial recognition has a long research history. It started to become popular in the early 1990s due to the introduction of the historical Eigenface approach [28]. The low-dimensional representation drove the early approaches through certain distribution assumptions, such as sparse representation [29], [30], linear subspace [31], [32], and manifold [33], [34]. Limited performance was achieved. In 2012, AlexNet was introduced in the ImageNet competition [35]. It applied deep learning in this field and significant performance improvement was accomplished. Number of variances deep

learning based facial recognition algorithms were developed. Convolutional neural networks obtained test results.

2.8 Facial Landmark Detection

Facial landmarks refer to points of human faces that mark important and identifiable features and specific points, such as the edges of the face, eyebrows, eyes, nose, lips, and chin on a face. By studying these points, the identity of the owner of the face can be known, the direction the person is facing can be calculated, and their facial expression and emotion can be estimated. Using such information, we can better monitor the status of the driver and other occupants. We can know if the driver is focusing on the road, chatting with other passengers, experiencing fatigue, etc. Facial landmarks detection can also be applied in different fields such as entertainment, security surveillance, and medical applications [36], [37], [38]. Deep learning-based approaches obtained state of art performance in this field such as [37], [39], [40].

In the paper [41], the authors selected eyebrows, eyes, lower nose, and mouth as the feature to be detected. The images are first transformed into grey images. Then, recursive Gaussian transformation is applied to smooth out the grey images. Finally, the landmarks are identified, and their orientations are calculated. The orientations of these facial features are used to detect the emotion shown on these faces, assuming different emotions lead to different facial feature orientations. The system is tested with the Pictures of Facial Affect database consisting of consists of 110 images of six males and eight females with three different resolutions. It achieved high accuracy.

With neural networks' increase in popularity, numerous research was conducted

on applying neural networks to facial landmarks detection. In 2018, Song, Kang, and Lee applied convolutional neural networks to the task of facial landmark detection [42]. The system consists of a global network and a local network. The global network is responsible for locating the landmarks sparsely in downsized images. The result is then fed to the local network. The local network then obtains the precise location of these landmarks. The network is trained with 29,000 images and tested with the 300W dataset and LFW dataset. The network detected 68 landmarks with relatively high accuracy. Deep learning-based model is used for facial landmark detection in this research.

2.9 Neural Network Quantization

Quantizing a neural network model is the process to reduce the precision of the weights, biases, and activations so that the model can consume less memory and use less computation power. The parameters in a neural network model are usually 32-bit float numbers. There are often millions of parameters in a model. Running such a model requires a powerful hardware environment. To reduce the cost, it is required to quantize the model by using a smaller representation, usually 8-bit integers. For example, reducing from 32-bit to 8-bit, the model size will be reduced by a factor of four. Quantization is explored in this research work to quantitate the trained models.

CHAPTER THREE

OBJECT DETECTION AND AUTO LABELING

3.1 Introduction

Object detection has important applications, such as driver monitoring, medical imaging analysis, video surveillance, and so on. Supervised deep learning neural networks such as You Only Look Once (YOLO) have demonstrated its superpower in this domain and been applied in real life successfully [43]. The deeper and the bigger the neural networks are, the more powerful they become. There can be millions and millions of trainable parameters in a deep learning neural network. To train such huge model, usually an enormous amount of training data is needed. Furthermore, the more varieties of objects are used to train a model, the better performance it can achieve. So, samples used during training process play a critical role to building a large neural network with high performance. However, human effort is usually needed to label the training data which can be expensive and time consuming.

Many datasets have been published to be used in different areas and meet various needs. But it is particularly challenging to find an existing public dataset that can satisfy specific and complex requirements completely. For example, Microsoft Common Objects in Context 2017 (COCO 2017) dataset is used by many researchers to train and test their object detection models [44]. It has 123,287 images and 886,284 instances that fall into 80 classes. However, many instances outside of these 80 categories, such as human faces, are not labeled. The usages of this dataset are limited by these 80 classes. It cannot be used to train a model for driver monitoring which requires human faces to be detected

along with other objects, such as seats, dogs, toys, baby seats, and so on, which commonly appear in the vehicle. To the best of our knowledge, a publicly available dataset which has enough samples and can be used in this scenario does not exist. Adding bounding boxes of faces to COCO 2017 dataset can satisfy the needs of building a driver monitoring model. So, it is desirable to customize COCO 2017 dataset to meet this requirement.

The Web Image Dataset for Event Recognition (WIDER) dataset is a publicly available static image dataset and used for complex event recognition [45]. A total of 32,203 images are selected from WIDER dataset which makes up the Web Image Dataset for Event Recognition FACE (WIDER FACE) dataset [46]. There are 61 event categories from real world miscellaneous scenes in the WIDER FACE dataset. Total 393,703 faces are labeled with bounding boxes and can be used for face detection. However, only faces are labeled in this dataset. So, it is not suitable to train a driver monitoring model which needs to recognize seats, pets, and other common objects in the vehicle either.

COCO 2017 and WIDER FACE contain all the classes needed to train a driver monitoring model. But they cannot be used together directly because the objects of the same class may exist in both datasets, and they are labeled differently. In the case, one object is labeled as positive in one dataset, but it is labeled as negative in the other one. The model will be confused by the conflicts. Manually labeling objects is expensive. Assuming manually labeling faces in one image in the COCO 2017 dataset takes two minutes on average. Finishing all 123,287 images in the COCO 2017 dataset, around 4110 hours are needed, which is very labor-intensive.

To reduce the cost and improve efficiency, a deep learning-based auto-labeling system is proposed in this paper. First, an Auto-Labeling Model (ALM) is trained using the WIDER FACE dataset which focuses on detecting face only. Second, the trained ALM is applied to COCO 2017 dataset and produces bounding boxes for faces automatically. Third, these face bounding boxes are merged with the existing labels in COCO 2017 by the system; COCO 2017 is expanded from existing 80 classes to 81 classes. Finally, the quantitative evaluation of labeling performance is conducted, and the experiment result indicates that the proposed method is dependable and feasible.

A YOLO based ODNN is built and trained using the expended dataset to detect the common objects in the vehicle.

3.2 Expand COCO Dataset Automatically

The auto-labeling system proposed in this research work is depicted in Figure 3. The system works as following: 1) ALM is trained using WIDER FACE dataset which can provide very accurate bounding boxes for faces; 2) The filter program automatically processes COCO 2017 dataset and only keeps the images with human object; 3) ALM detects faces on the filtered images and provides corresponding bounding boxes; 4) the label processing program automatically merge the labels provided by ALM to COCO 2017 dataset; 5) Finally the COCO 2017 dataset is expanded from 80 classes to 81 classes.

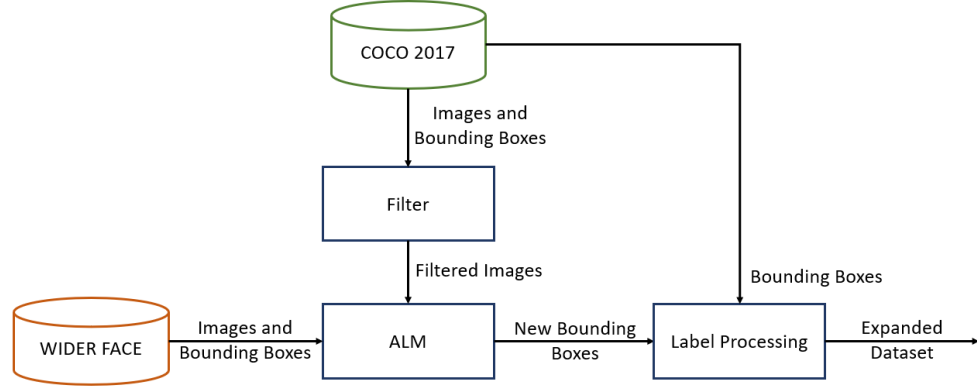


Figure 3. Auto-labeling system.

3.2.1 Auto-Labeling Model

A YOLO based ALM model is built and trained in the experiment to provide the bounding boxes automatically for human faces in COCO 2017 dataset. The overall structure of ALM is depicted in Figure 4, consisting of a total of 64 layers including 61 convolutional layers, a max pooling layer and 2 up sampling layers. Each color bar in the figure represents each layer, and the numbers from 0 to 63 on top of each bar represent each layer. Sigmoid activation function is applied to each convolutional layer except for the last three layers. There are 7,228,582 trainable parameters in total. Adam optimizer with learning rate of 0.01, coefficients of (0.937, 0.999) and weight decay of 0.005 is used. Anchor sizes of (10,14), (16,31), (34, 24) (31,60), (63,46), (60,119), (62,44), (59,119) and (373, 326) are used.

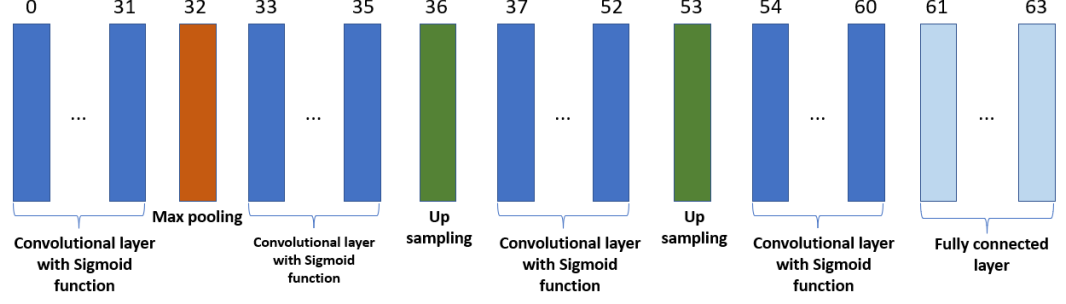


Figure 4. The structure of ALM.

Three loss values are used to calculate the overall loss. The three losses are coordinate loss L_{co} , classification loss L_{cl} and objectness loss L_{ob} . L_{co} , measures the distance error between of the label box and predicted box, L_{cl} measures the error between the actual class and the estimated class, and L_{ob} measures if a bounding contains an object. Squared error $(Y(x_i) - y(x_i))^2$ is used to calculate L_{cl} where x_i is the input, $Y(x_i)$ is the actual class and $y(x_i)$ is the estimated class. L_{co} equals to the error between 1.0 and the $IOU(x_i)$ of the object detected, where x_i is the input.

To address the extreme foreground-background class imbalance issue during training, Focal Loss function was designed by Lin et al. to improve the performance of one-stage object detection model in their research work [47]. When it comes to face detection, faces of large scales are easy to distinguish and faces of small scale are hard to detect from their background. Focal Loss listed in equation 1 is used to calculate objectness loss L_{ob} in this research. Instead of using the ground truth label value $Y(x_i)$ in Cross Entropy loss listed in equation 2, Focal Loss function uses a factor of $(1 - Y(x_i))^r$ to balance loss between the hard and easy samples. $Y(x_i)$ is the predicted

probability, x_i is the input, and r is the modulating factor, which is 1.5 in this experiment.

The overall loss $L(x_i) = L_{co}(x_i) + L_{cl}(x_i) + L_{ob}(x_i)$ is used to update weights and bias.

$$L_F(x_i) = -(1 - Y(x_i))^r \log(y(x_i)) \quad (1)$$

$$L_E(x_i) = -Y(x_i) \log(y(x_i)) \quad (2)$$

WIDER Face dataset is fed to ALM for training. The training samples are listed in Figure 5. Faces with different backgrounds, sizes, angles, with or without glasses, with or without hats and so on are labeled by bounding boxes. Not only are full faces labeled but partially blocked faces are also labeled. One image can include one face or multiple faces. The variety and richness of the training samples ensure the performance of trained model. A total of 300 epochs are trained on 2 NVIDIA GeForce GTX 1080 Ti GPUs and data parallelization is adopted. The training progress is displayed in Figure 6. All three training losses decreased significantly after few training epochs and continued decreasing slightly for the rest of the training epoch.

F1 confidence curve, precision confidence curve, recall confidence curve and precision recall curve are depicted in Figure 7. The validation examples are listed in Figure 8. These two figures indicate that the model is well trained. Faces with large scales, small scales, different poses, partially hidden, and so on are well detected. The bounding boxes are accurately located at the appreciate positions.



Figure 5. Training samples of WIDER FACE dataset.

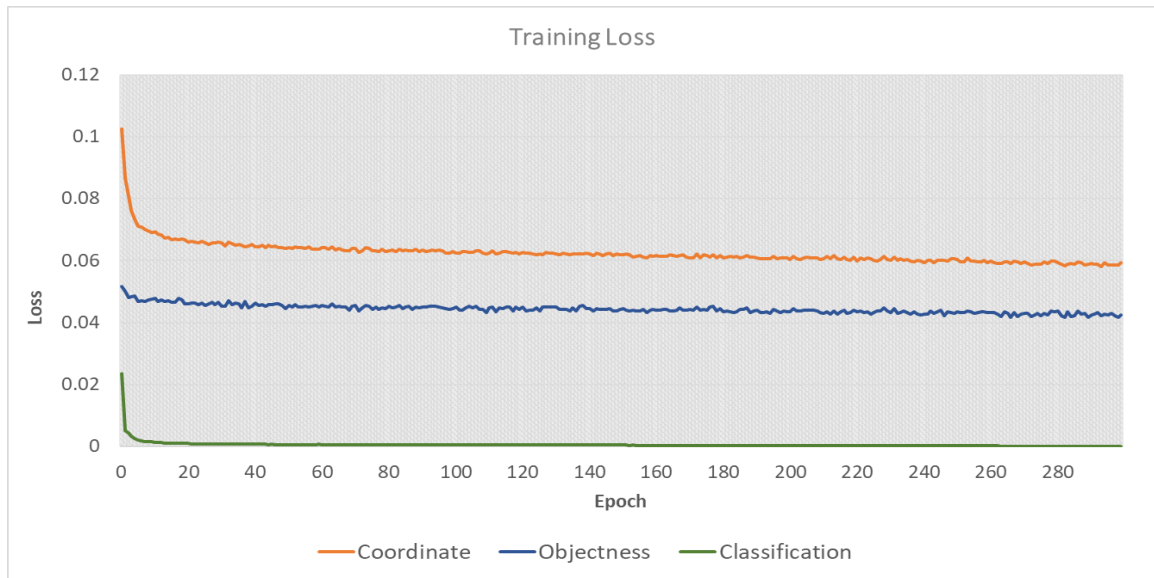


Figure 6. ALM training loss.

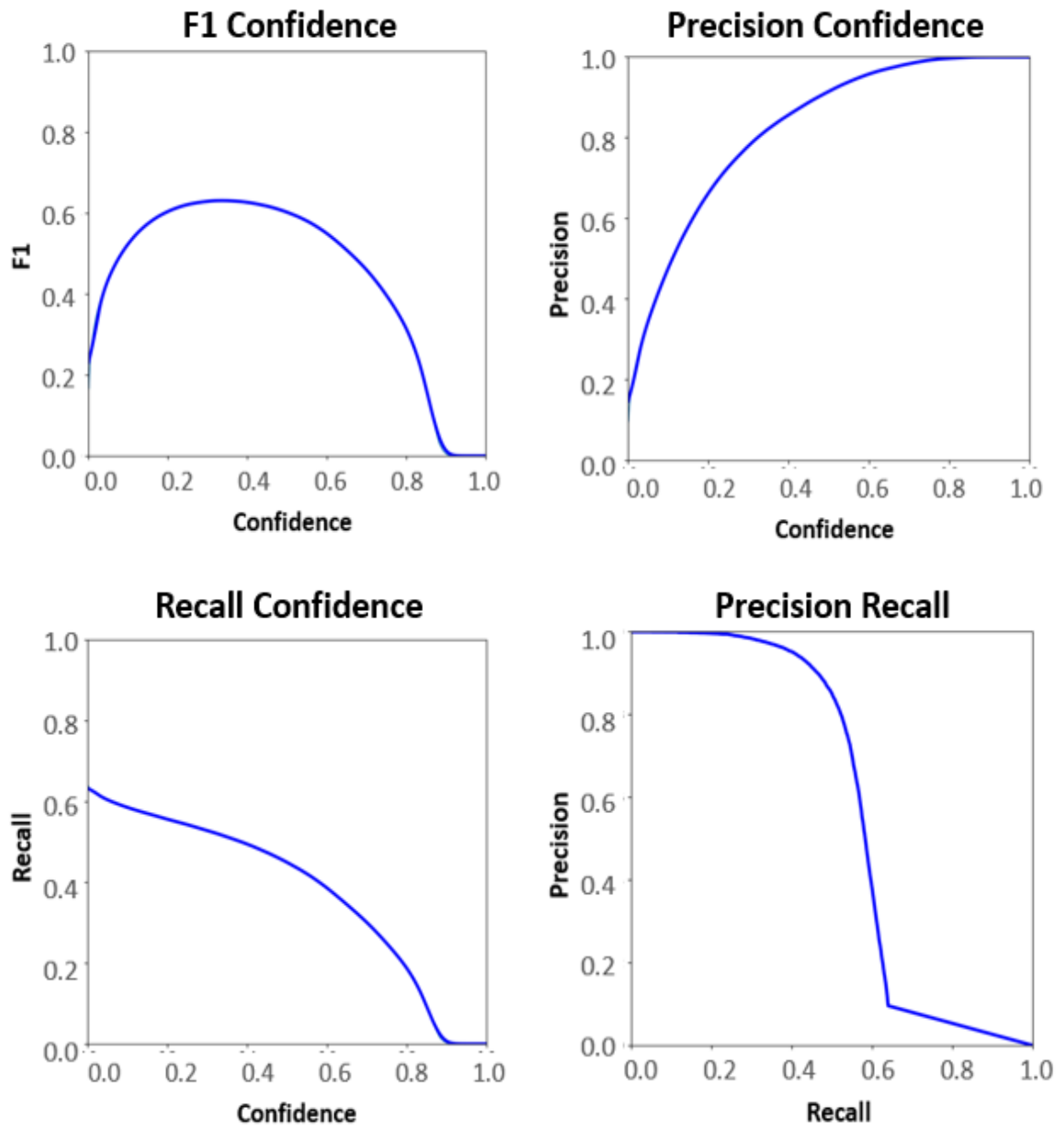


Figure 7. ALM performance



Figure 8. ALM validation examples

3.2.2 Label Automatically

The trained ALM is applied to COCO 2017 dataset to detect faces. To exam the performance of ALM, all images in COCO 2017 are used during evaluation. Figure 9 lists the examples of detection results. Total 54,841 images in COCO 2017 are labeled with human objects and these images are used for face detection. 205,065 objects are classified as faces by ALM. In the scenario of driver monitoring, it is not necessary to detect faces with small size. Any detected bounding boxes which width $\times$ length < 400 are eliminated automatically by a program in the system. Total 34,789 images and 60,941

objects classified as faces remained. Around 12 minutes are used by auto-labeling system in total, but 4110 hours are needed to manually label all these images.



Figure 9. Examples of faces detected by ALM in COCO 2017.

3.2.3 Labelling Results

There is no existing method to automatically exam ALM detection results with zero tolerance. To evaluate the effectiveness of ALM, these detected 60,941 faces after eliminating small sizes are visually verified one by one. Examples of objects that are detected wrongly as human faces in COCO 2017 are listed in Figure 10. Cats and dogs

are wrongly identified as human faces indicates that WIDER FACE training dataset lacks such negative samples. This can be addressed in future study along with other mistakenly detected objects. 58,740 valid bounding boxes out of detected 60,941 bounding boxes of faces are kept. The effectiveness rate is 96.4%.

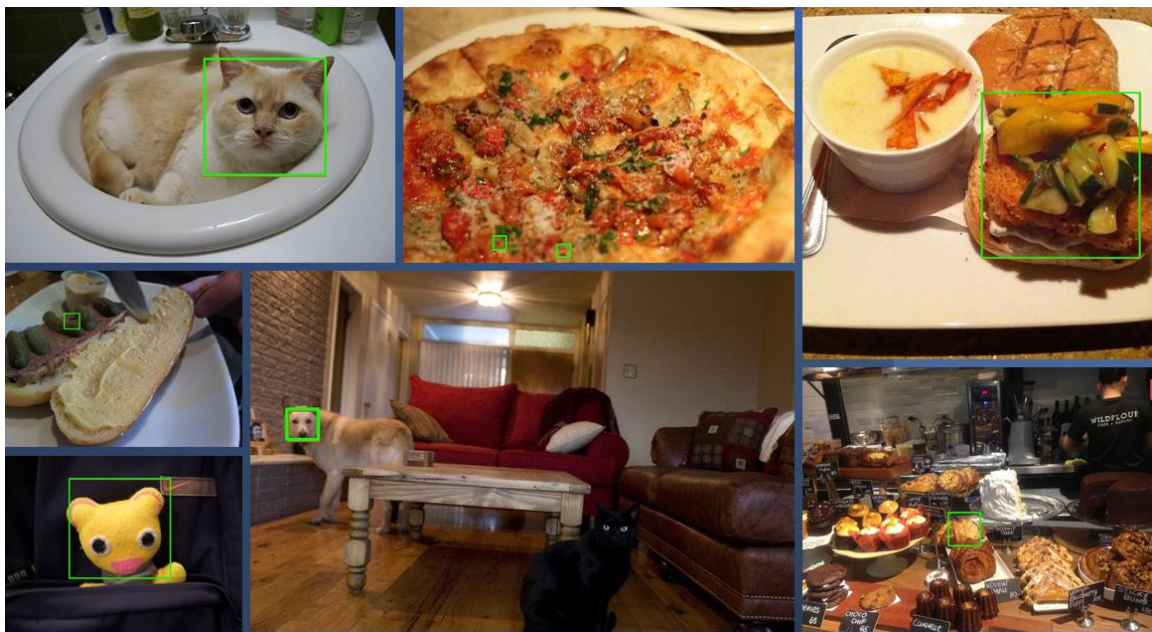


Figure 10. Examples of wrongly detected as faces by ALM in COCO 2017.

3.3 COCO Dataset Expansion and Object Detection

These valid faces corresponding to bounding boxes detected by ALM are merged with COCO 2017. COCO 2017 dataset is expanded from 80 classes to 81 classes. Images without human faces are also eliminated. In the expanded dataset, there are total 33,000 images, and 352,485 objects distributed in 81 classes. There is average 4,352 objects per class and average 11 objects per image. Please see the objects distribution in Figure 11. People take 34% and faces take 17% in our new dataset which makes our ODNN easy to

detect them. Other common objects such as cell phones, pets, foods and so on are distributed evenly.

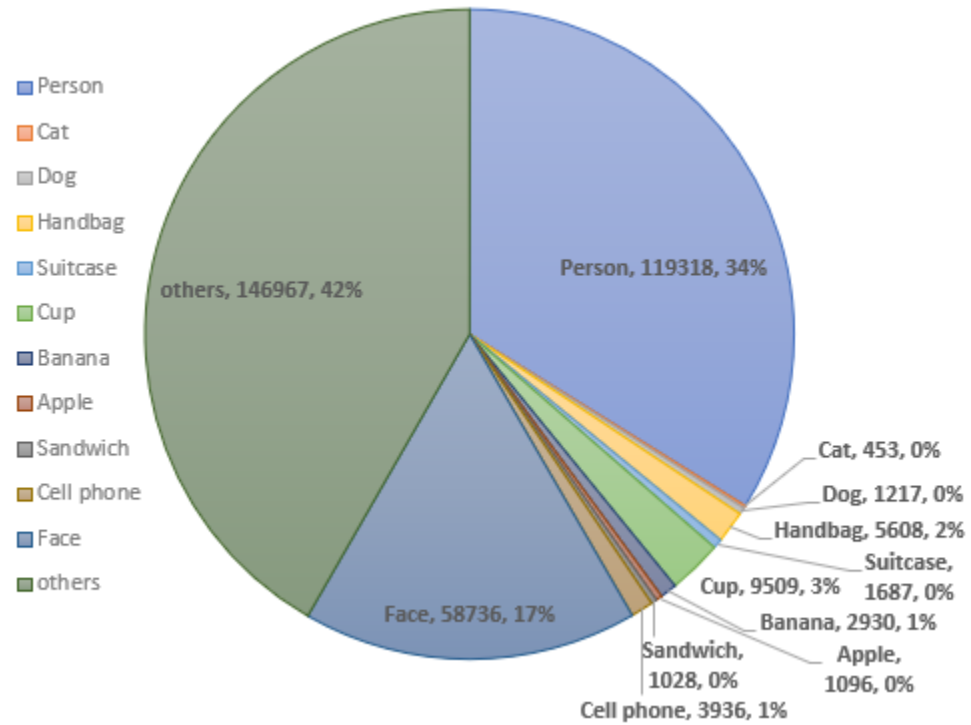


Figure 11. Distribution of expanded object detection dataset

A new YOLO-based ODNN is trained using this new dataset. The detection examples are listed in Figure 12 and validation results are displayed in Figure 13. Faces are well detected along with other existing classes in the original dataset. The performance of faces detection is evaluated and presented by F1 confidence curve, precision confidence curve, recall confidence curve and precision recall curved. Across all four curves, face detection results are above the average level of all classes. This

indicates that the auto-labeling system can provide accurate bounding boxes to the original dataset and the model trained on the new COCO 2017 dataset is effective.

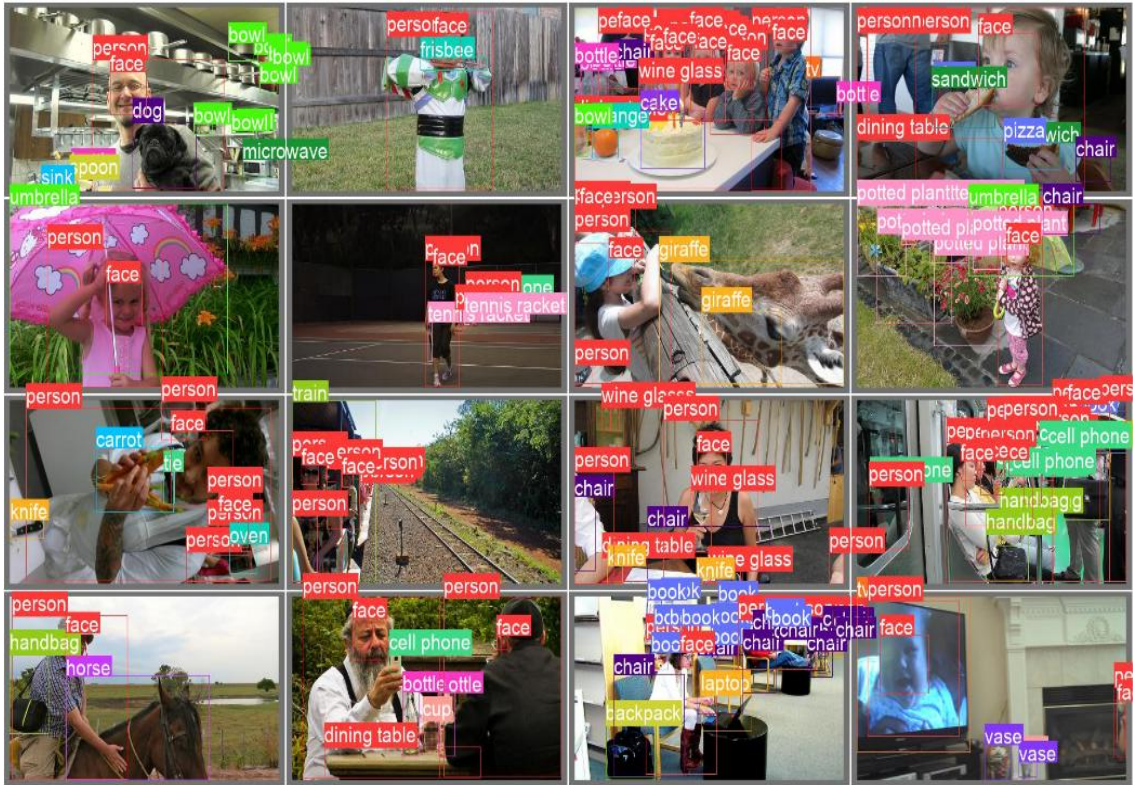


Figure 12. Detection results of the expanded dataset.

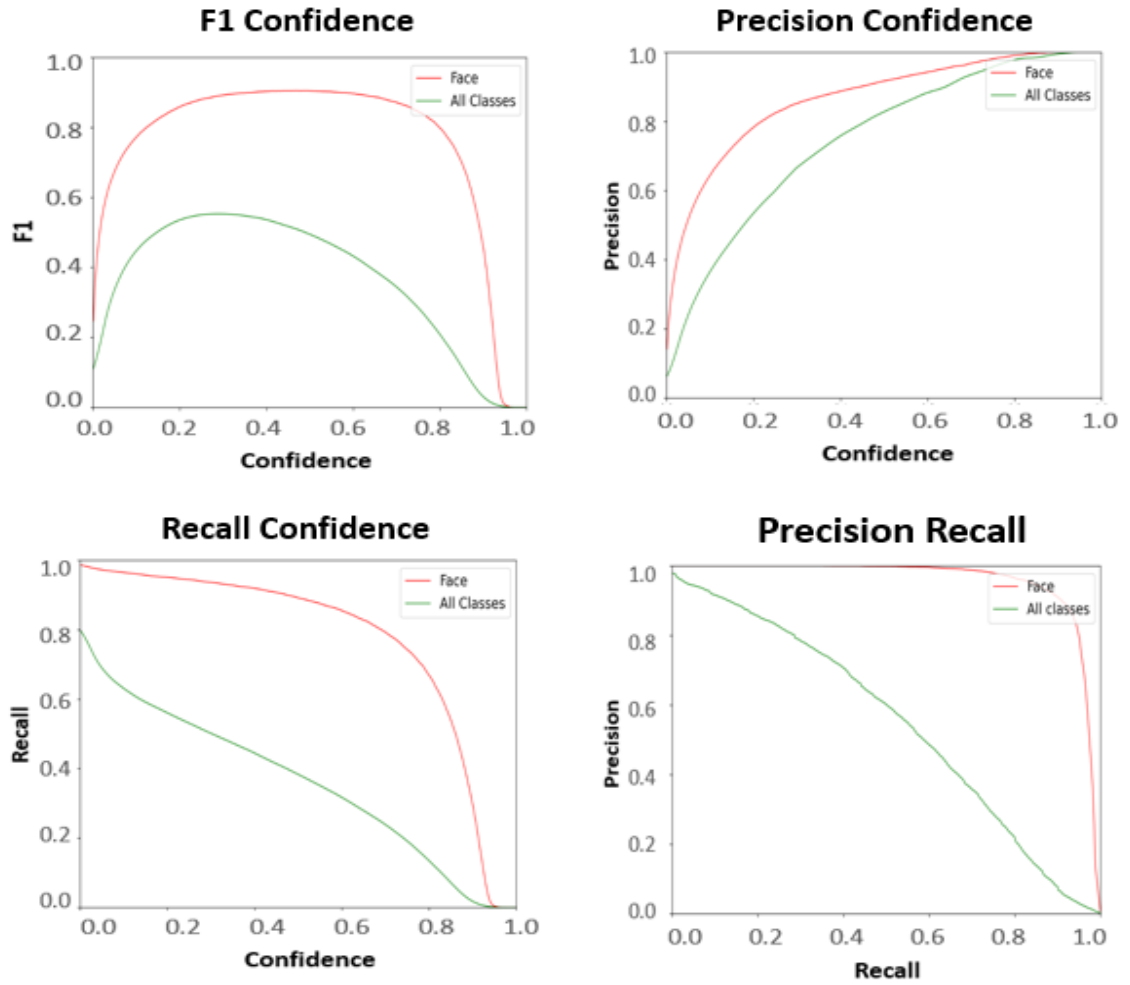


Figure 13. Object detection performance on the expanded dataset.

3.4 Summary

A bounding box auto-labeling system is proposed in this research work. In the system, a YOLO based Auto-Labeling Model (ALM) is trained using WIDER FACE dataset. ALM can provide accurate labels of the faces in COCO 2017 dataset. The COCO dataset 2017 is expanded from 80 classes to 81 classes by adding human face class automatically. The cost is extremely low, and 4110 hours of human effort is saved. The expanded dataset is evaluated qualitatively and examined visually and proved that the

proposed system is effective. The idea proposed in this study can be extended to other similar datasets and save significant amounts of human effort.

ODNN detects common objects in vehicles such as humans, faces, baby seats, pets, cell phones and so on. The detected objects are the input of FRNN and FLDNN.

CHAPTER FOUR

FACIAL RECOGNITION

4.1 Introduction

Many researchers have developed different models for facial recognition [31], [33], [36], [38], [48], [49]. The model sizes range from a few megabytes to half gigabytes. The smaller size of the model is preferred in our research. For example, MobileNets [50] was published by Google in the field of embedded applications. It has been widely used for image processing.

4.2 Technical Approach

Residual neural networks were introduced by He et al. in the reach work of image processing [51]. In a large multi-layer deep learning neural network, there are many layers stacked together. These layers can be grouped together to form subnetworks denoted as $H(x)$ where x is the input of a subnetwork. The subnetwork is re-parameterized by residual learning. The parameter layers are represented as a residual function denoted as $F(x) = H(x) - x$. The output y of the subnetwork is represented as $y = F(x) + x$. A skip connection implements the operation of “+”. This skip connection is identified as the mapping from the input to the output of its subnetwork which is called residual connection. Previous research in [51] has proved that this residual connection can stabilize the training process and convergence of deep neural networks quickly which has hundreds of layers.

An example of residual neural networks is depicted in Figure 14. The residual block takes x as the input and x is connected to the output of y at the end after a few layers. They are combined as the input of next layer or block. The layers between the input and output in a residual block can be different number of layers and different structures.

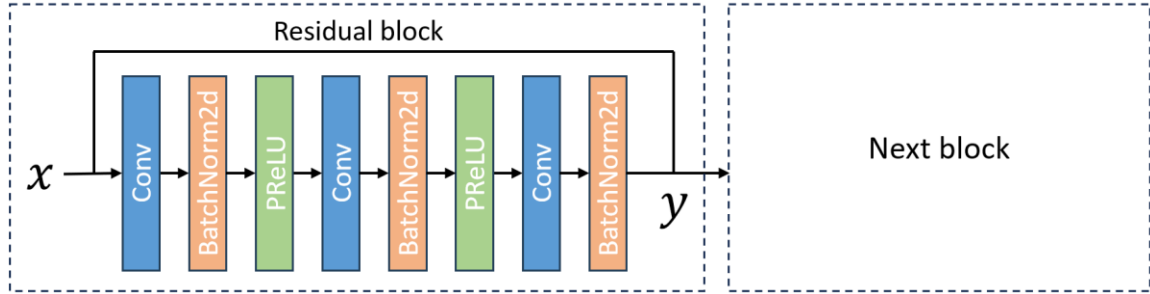


Figure 14. Residual neural networks.

To build a model that can be used in a low-cost hardware environment with limited memory and computation capacity, Depthwise Separable Convolution (DSC) with residuals is adopted in this research to build FRNN in the residual blocks. Many researchers have been focusing on increasing accuracy when building deep convolutional neural networks which require high end computing hardware. It is not feasible to reference large models in the embedded environments. So, it is desired to reduce the model size and increase the reference speed but without losing much accuracy. DSC was initially introduced by Laurent et al. in the research of image texture classification [52]. It was experimentally proven that DSC could achieve similar performance but with much smaller size and faster speed. It has been applied in different fields such as neural

machine translation[53], plant disease classification[54], semantic segmentation in street–road scenes[55], high-speed synthetic aperture radar ship detection[56] and so on.

A standard convolution filters the input channels and combines the filter results into a new set of outputs in one step. An example of a standard convolution is depicted in Figure 15. It takes the feature map with size of $H \times W \times D$ as the input tensor, applies N kernels with size of $K \times K \times D$ using stride of 1 and no padding to generate N output tensors with size of $(H - K + 1) \times (W - K + 1) \times 1$. The standard convolution cost is:

$$\text{Cost}_{std} = H \cdot W \cdot K^2 \cdot D \cdot N \quad (3)$$

The calculation cost of multiplications is very expensive especially in the embedded system. So, it is desire to reduce the number of multiplications.

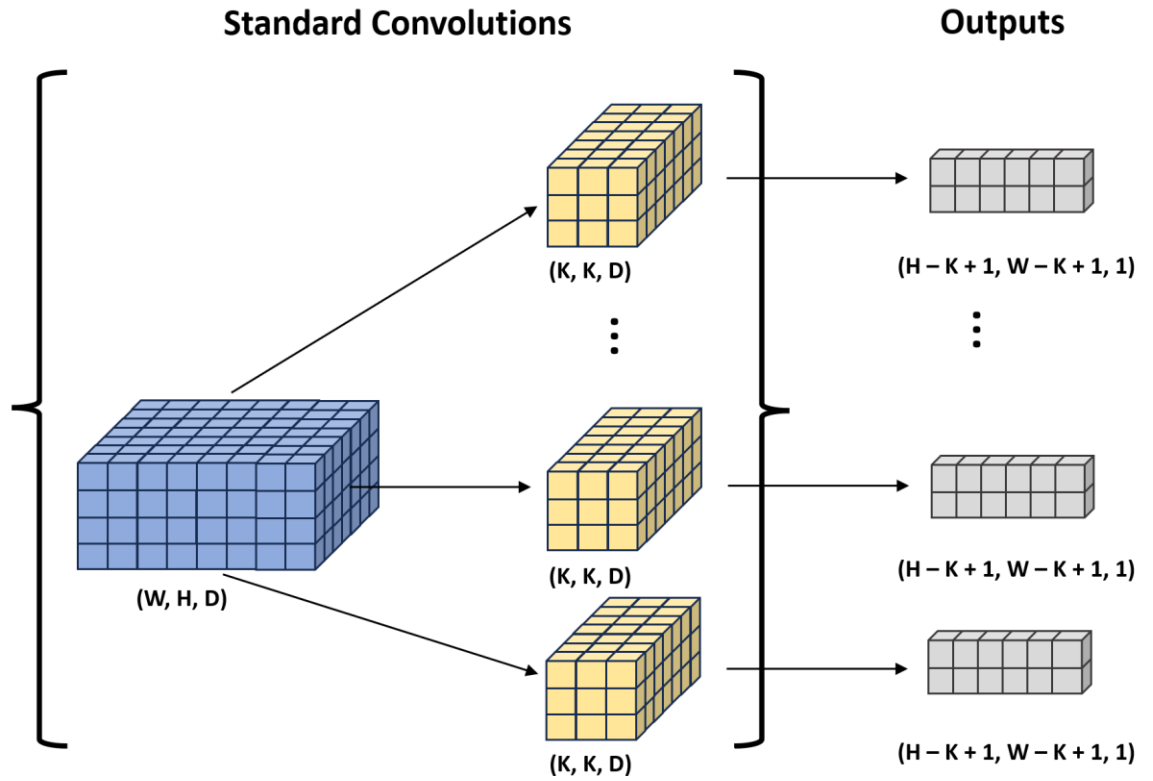


Figure 15. Standard convolution.

Compared to the standard convolution, DSC is a form of factorized convolutions. Instead of producing the output in one step, it factorizes the calculation into two steps: a depthwise convolution and a pointwise convolution. First, a single filter is applied to each channel of the input tensor using depthwise convolution, then a pointwise convolution is used to combine the depthwise convolution results to produce the final output. Instead of using only one layer in standard convolution, DSC has two layers: a depthwise convolution layer is used for filtering and a pointwise convolution layer is used for combining. Both computation and model size are drastically reduced by factorization.

An example of DSC is depicted in Figure 16. The depthwise convolution layer takes the input tensor with size of $H \times W \times D$ then applies convolution by a kernel with size of $K \times K \times D$. Instead of calculation the full convolution, depthwise convolution layer only calculate the convolution for each channel and does not combine the convolution results from each channel. It generates interim results with size of $(H - K + 1) \times (W - K + 1) \times D$. The depthwise convolution layer is followed by a pointwise convolution layer which combines the interim results using kernel size of $1 \times 1 \times D$. Finally, DSC produces the tensors with size of $(H - K + 1) \times (W - K + 1) \times D$. The DSC cost is:

$$\text{Cost}_{dsc} = H \cdot W \cdot D \cdot (K^2 + N) \quad (4)$$

The reduction of computation cost is:

$$\frac{\text{Cost}_{dsc}}{\text{Cost}_{std}} = \frac{H \cdot W \cdot D \cdot (K^2 + N)}{H \cdot W \cdot K^2 \cdot D \cdot N} = \frac{1}{N} + \frac{1}{K^2} \quad (5)$$

The calculation cost saved is the factor of K^2 approximately. Almost 9 times computation cost is saved if the kernel size of 3 is used. The experimental results indicate that there is only a very small reduction in accuracy.

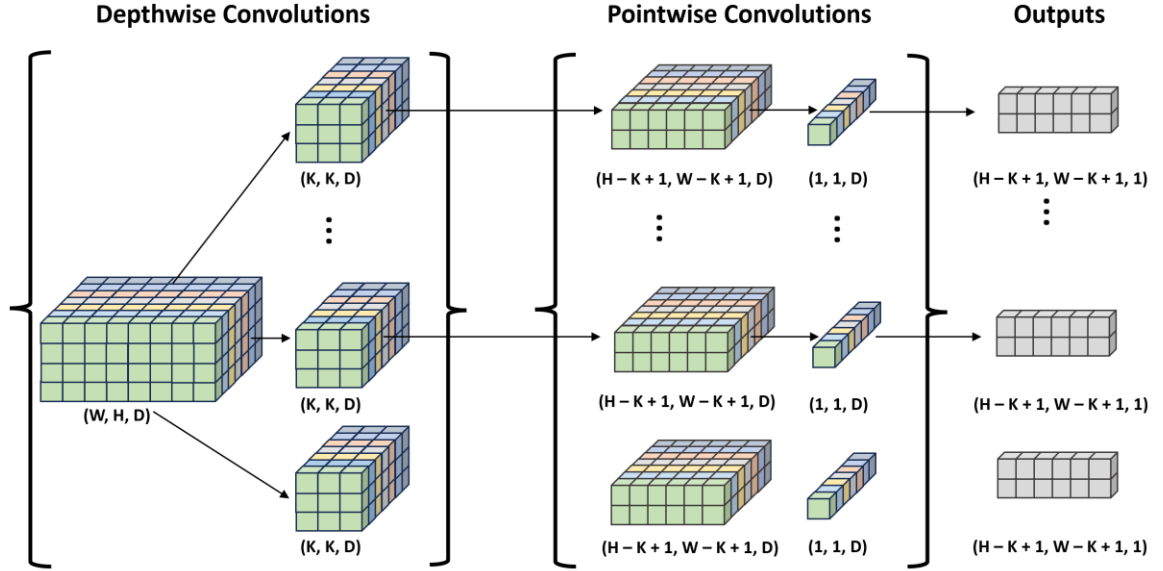


Figure 16 Depthwise separable convolution.

4.2.1 FRNN Structure

The basic building blocks of FRNN are sequential and residual blocks with depthwise separable convolution depicted in Figure 17. It starts with a sequential block, then followed by 15 inverted residual blocks. Another sequential block after these inverted residual blocks produces the facial recognition result feature map which is used by the software in DDMS for driver ID identification. The starting sequential block two sub-blocks have a convolutional layer and a batch normalization layer. Parametric ReLU (PReLU) activation function is used in each sub-block.

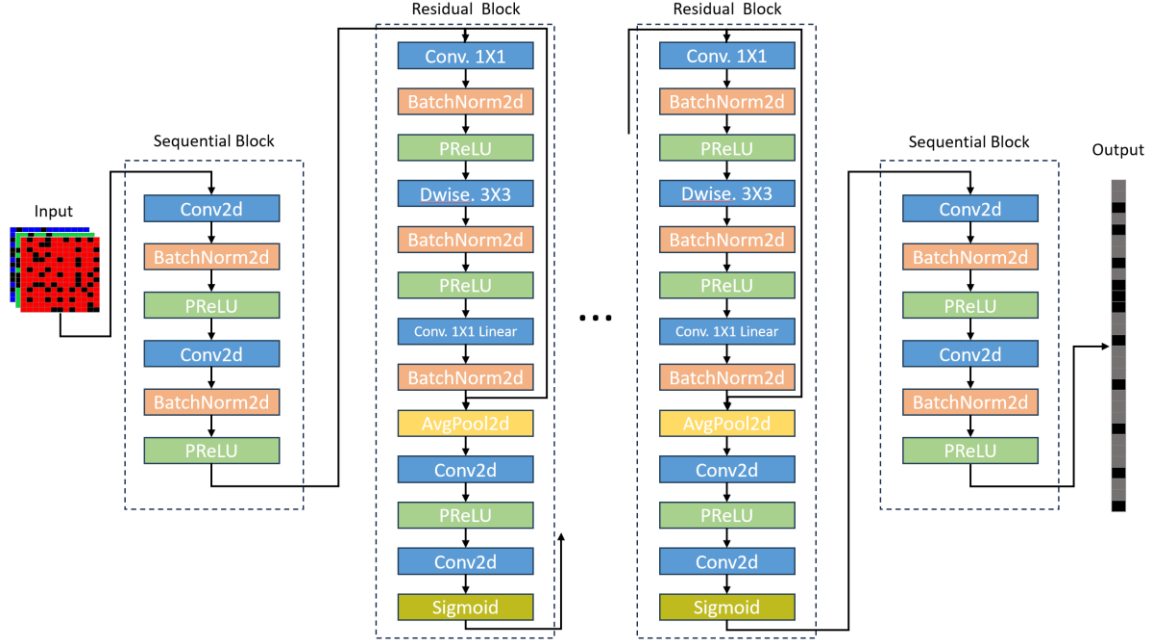


Figure 17 FRNN architecture design.

Compared to Leaky ReLU $LReLU(x) = \max(0, x) + A * \min(0, x)$ which uses a predefined negative slope value. PReLU listed below with trainable parameter a can learn the slope during training process adaptively. Without using fixed slope value, the most appropriate slope of each neuron in the neural network can be learned using PReLU. So, the capacity of the model capturing complex patterns can be potentially enhanced. PReLU is used throughout FRNN design.

$$PReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ ax, & \text{otherwise} \end{cases} \quad (6)$$

Inside of residual blocks, DSC is the basic network structure. It starts with a convolutional layer; a bath normalization layer follows after that; PReLU calculates the output of this small block; a depthwise convolution layer takes the output from the

previous small block; a batch normalization and PReLU activation function are applied; another convolutional layer and a batch normalization layer are the last two layers in a residual block. The input x of a residual block is combined with y the output of last. The result of $x + y$ are the input of next block. The total numbers of trainable parameters are in FRNN 1,116,478.

4.2.2 Facial Recognition Dataset

VGGFace2 [57] is used to train FRNN which can be downloaded from <https://www.kaggle.com/datasets/hearfool/vggface2>. Total 3,141,886 images are in the training dataset, and 169,396 images are validation dataset. These images are from 9,294 individuals including different genders, ages, ethnicities, poses and so on, average 356 images per object. Examples of images in VGGFace2 are list in Figure 18. Each face is in the center of the image, with or without glasses, with or without makeup. The genders are balanced in this dataset which 40.7% are females. These people are from different occupations such as sports, politicians, actors, actresses and so on. Face recognition model trained on datasets with varieties of poses can provide higher accuracy when the faces with different poses. Similarly, there are different facial expressions in the images which also help to improve the mode accuracy in the real scenario.

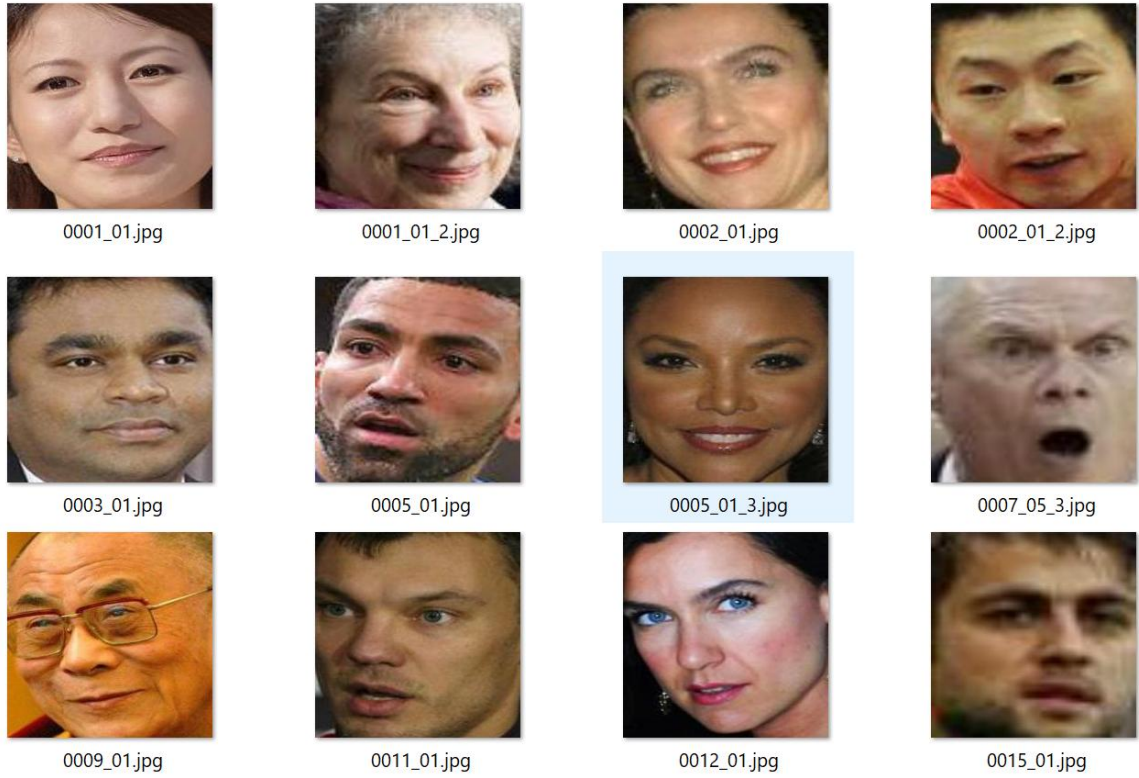


Figure 18 VGGFace2 example images.

4.2.3 FRNN Training

SoftMax cross-entropy loss is also called SoftMax loss which is commonly used in machine learning to optimize the models such as multi-classes classifiers. Facial recognition falls in this category and SoftMax cross-entropy is used to optimize FRNN model during training process.

Assume we have an input facial image x and p is the true label of x . Then we get a data point (x, p) for x and p . During training, there are K possible labels for each data point (x, p) . Our FRNN does not output the predicted probability $\hat{p}(k)$ of each class directly, instead, FRNN produces an estimated score $\hat{y}(k)$ of each class. The estimated

score is passed to a SoftMax function $\hat{p}(k) = \frac{\exp(\hat{y}(k))}{\sum_{k'=1}^K \exp(\hat{y}(k'))}$ to convert the K scores into a probability distribution over $\{1, \dots, K\}$. Cross-entropy loss function $L(p, \hat{p}) = -\sum_{k=1}^K p(k) \log \hat{p}(k)$ is applied to these to probabilities distribution $\hat{p}(k)$ to calculate the errors between the true value p and estimated scores $\hat{y}(k)$.

$$\begin{aligned}
L(p, \hat{p}) &= -\sum_{k=1}^K p(k) \log \hat{p}(k) \\
&= -\sum_{k=1}^K p(k) \log \left[\frac{\exp(\hat{y}(k))}{\sum_{k'=1}^K \exp(\hat{y}(k'))} \right] \\
&= -\sum_{k=1}^K p(k) \hat{y}(k) + \log \left[\sum_{k'=1}^K \exp(\hat{y}(k')) \right] \quad (7)
\end{aligned}$$

SoftMax loss function implicate assumes that SoftMax is used to calculate the model scores then cross entropy loss function takes the scores as its input. This is a quite common combination in machine learning, and all the deep learning frameworks provide such function by default without asking the users to convert the scores into class probabilities before the loss is calculated.

Stochastic gradient descent algorithm (SGD) is used to train the model with initial learning rate of 0.4 and momentum of 0.9. Compared to SGD without momentum, using SGD with momentum can accumulate the training speed in the consistent directions and the training loss converges much faster as well. Furthermore, the learning rate decays with factor of 0.0001 at the milestones of 3, 6, 9 and 13 epoch during training. Total of 40 epochs are trained on 2 NVIDIA GeForce GTX 1080 Ti GPUs displayed in Figure 19. Total around 20 hours is used, and data parallelization is adopted during training process. The training progress is displayed in Figure 20. The accuracy increases rapidly to around

95% in the first 3 hours then increases adjectively for the rest of the time with help of SGD with momentum and decaying learning rate. The accuracy reaches 96.8% at the end.



Figure 19. Training environment.

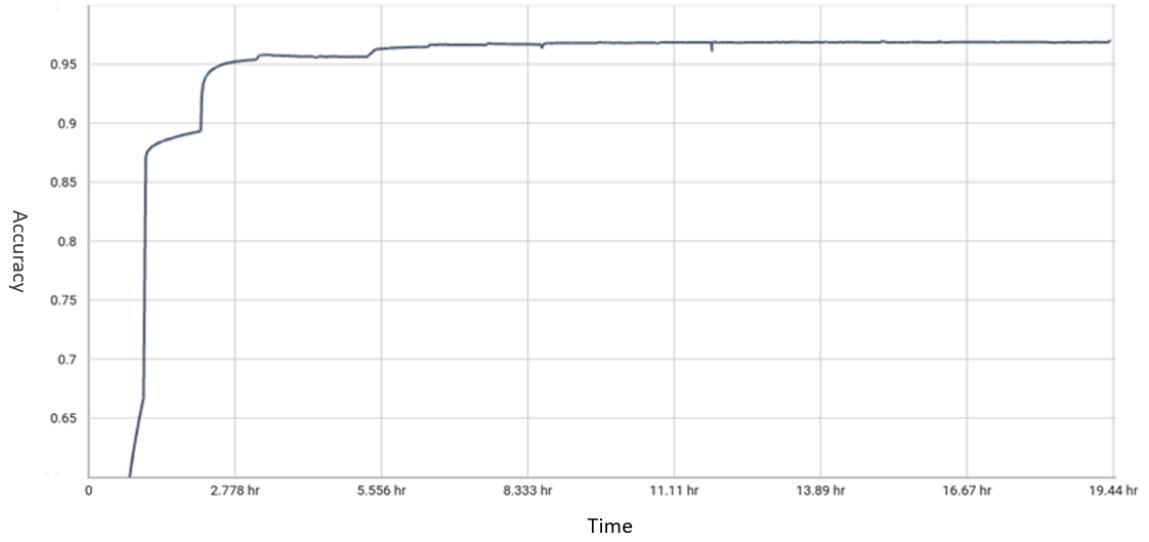


Figure 20. Accuracy increases as training time elapses.

4.3 Experimental Results

4.3.1 FRNN Performance

ROC is applied on the VGGFace2 test dataset to evaluate FRNN performance which shown in Figure 21. The area under the ROC curve is used to measure a classifier's ability. It distinguishes between classes across all thresholds, ranging from the random guessing to the perfect discrimination. The bigger the area under the ROC curve is obtained, the better performance FRNN has. The area is near to 1.0 which indicates that FRNN is very well trained.

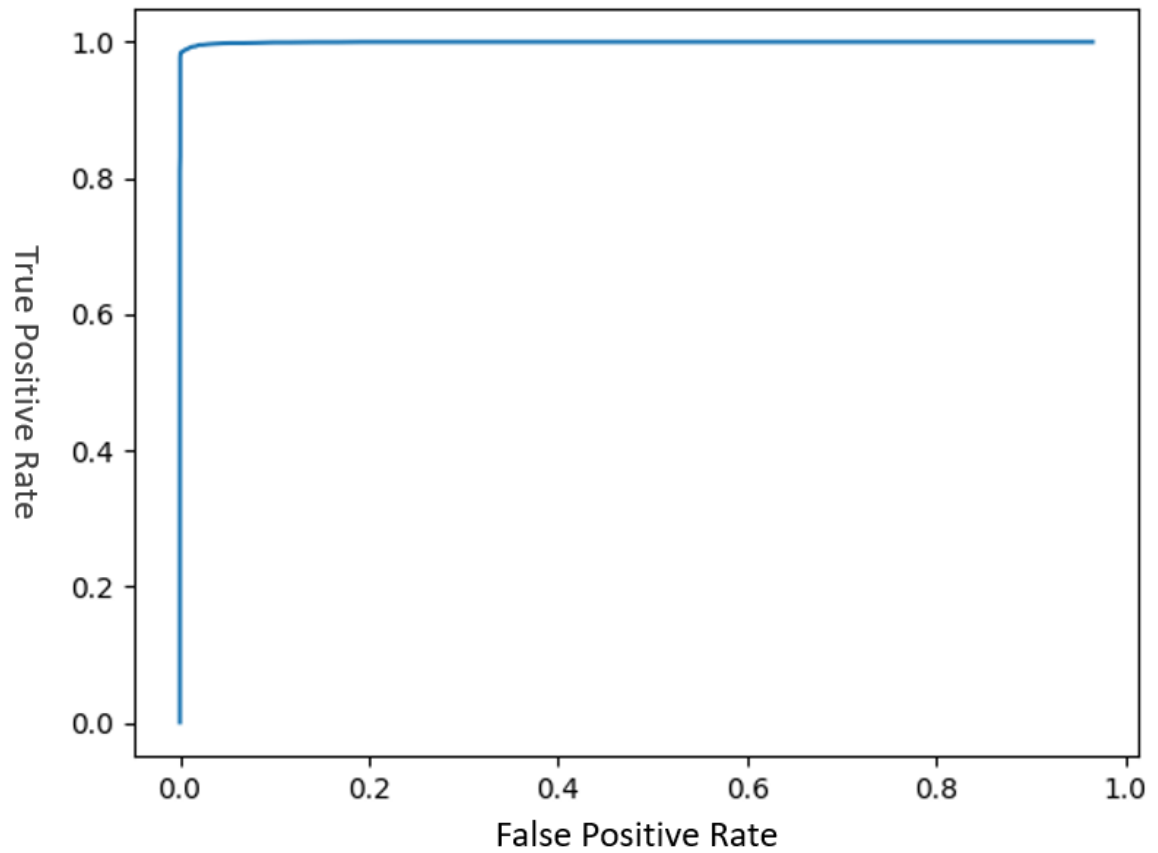


Figure 21. ROC of FRNN.

The trained FRNN is also applied to the images which unseen during training. Please see the examples of facial recognition results in Figure 22. All faces are correctly identified.



Figure 22. Facial recognition results.

4.4 Summary

The basic building blocks in FRNN architecture design are sequential and residual blocks with depthwise separable convolution. Parametric ReLU activation function is used in each block. SoftMax loss function is used when training FRNN on Based VGGFace2 dataset. The trained FRNN is quantitatively evaluated by ROC scores. The experiment result indicates that FRNN can recognize the faces well.

CHAPTER FIVE

FACIAL LANDMARK DETECTION

5.1 Introduction

Facial landmark detection is the process of detecting and localizing specific points or landmarks such as the eyes, nose, lips, and chin on a face. It can be used for facial expression analysis, head pose estimation, estimate gaze direction and so on. Some research works [37], [39], [40], [41], [41], [58] have been conducted by scholars. In this paper, FLDNN is built to detect facial landmarks.

5.2 Technical Approach

Deep learning-based approaches have been studied in recent years. A convolutional deep learning neural network is built in this research to implement this function. Depthwise separable convolution listed in Figure 16 is the basic building blocks in FLDNN.

5.2.1 FRNN Structure

FLDNN mode has similar structures to FRNN. The basic building blocks of FLDNN are sequential and residual blocks with depthwise separable convolution depicted in Figure 17. It starts with a sequential block, then followed by 7 inverted residual blocks. Another sequential block after these inverted residual blocks produces the detected facial landmarks which are used by the software in DDMS for driver status estimation. Similarly, parametric ReLU (PReLU) activation function at page 38 is also used in each block.

5.2.2 Facial Landmark Detection Dataset

Dataset with different number of key points of facial landmark are available. To achieve the best performance, the dataset with largest number of key points is used. There are different facial landmarks datasets available such as Labeled Face Parts in-the-Wild (LFPW) [59], Annotated Faces in the Wild (AFW) [60], iBUG Eye Segmentation (IBUG) [61] and so on. Each of these datasets has 68 points annotated on each image. To achieve better facial landmarks detection results, the Wider Facial Landmarks in the Wild (WFLW) dataset is selected in this research which can be downloaded from <https://wywu.github.io/projects/LAB/WFLW.html>. WFLW contains 7500 training and 2500 testing images with 98 annotated landmarks. This database has images with varieties of head pose, with or without make-up, different illumination, with or without blurring and different facial expressions and so on. Examples of the images in WFLW dataset are listed in Figure 23.

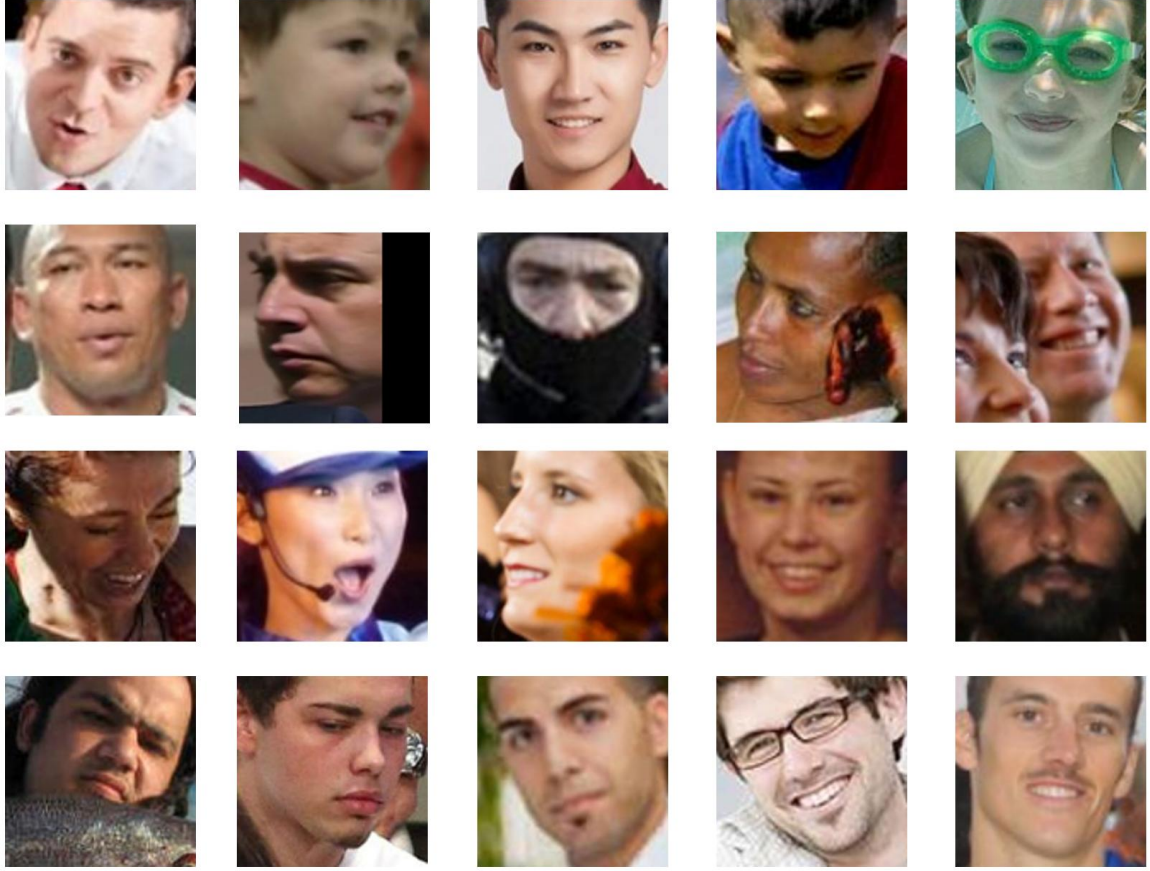


Figure 23 Examples of WFLW dataset.

5.2.3 FDLNN Training

A total of 500 epochs are trained on 2 NVIDIA GeForce GTX 1080 Ti GPUs and data parallelization is adopted. The training progress is depicted in Figure 24. Both training and validation losses drop quickly and reach under 0.5 at the end.

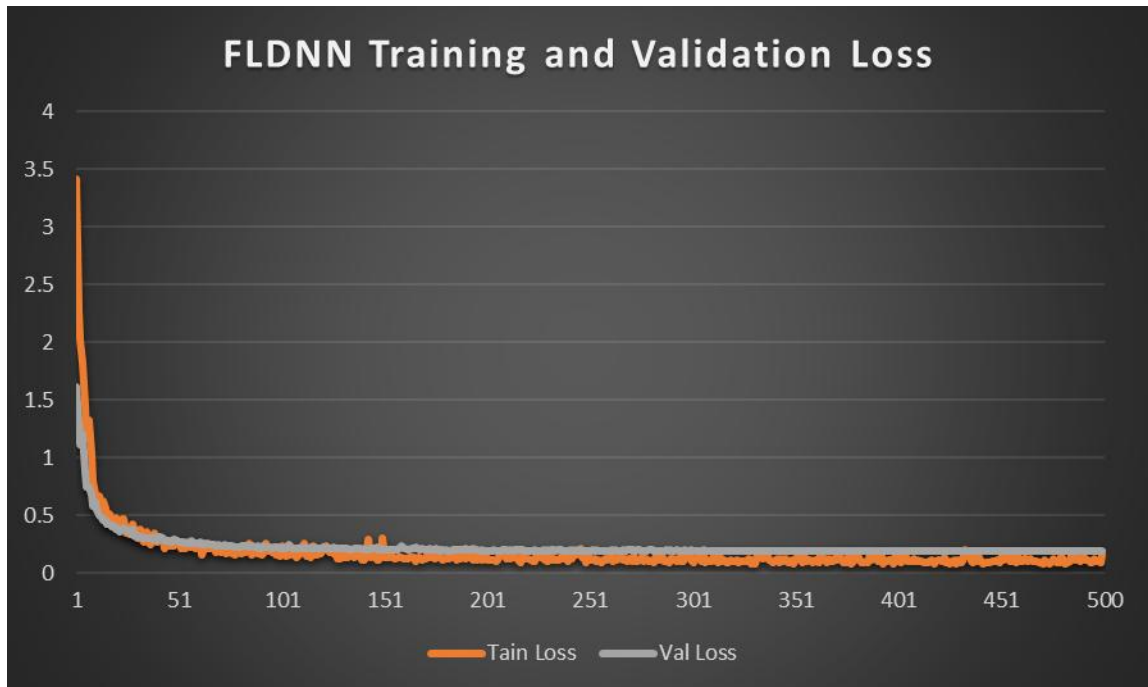


Figure 24 FLDNN training progress.

5.3 Experimental Results

Please see blow pictures of facial landmark detection results on the validation dataset in Figure 25. Facial landmarks are detected accurately on images with different poses, with or without glasses, with and without hats, different illumination, and including faces are partially occluded. FLDNN is also tested along with object detection and facial recognition results. Facial landmarks are detected nicely as well in Figure 26. Keypoints are accurately marked on the detected faces in the real-world environment.



Figure 25 FDLNN detection results of validation dataset.



Figure 26. Facial landmark detection result.

5.3.1 FDLNN Performance

The FDLNN performance is also evaluated quantitatively. 97.34 is achieved on the validation dataset.

5.4 Summary

The basic building blocks of FLDNN are sequential and residual blocks with depthwise separable convolution in FLDNN architecture design. Parametric ReLU activation function is used in each block. SoftMax loss function is used when training FLDNN on Based WFLW dataset. The trained FRNN is quantitatively evaluated by accuracy. The experiment result indicates that FLDNN can detect the key points well on images which are unseen during training as well.

CHAPTER SIX

NEURAL NETWORK QUANTIZATION

6.1 Introduction

Neural network quantization means performing computations and storing parameters in the model at lower bit-widths such as 8-bits rather than using floating point precision. After quantization, the model executes some or all the operations on tensors using integers instead of using floating point numbers. The size of the model can be reduced by the N times which $N = \frac{W_F}{W_I}$ where W_F bit-widths of floating-point number before quantization and W_I is the bit-widths of integer numbers after quantization. The memory needed during referencing process is much lower and the speed can be increased due to the faster computation of integer numbers. Additionally, the sacrifice of quantized model accuracy is usually limited.

The quantization function is defined in equation 8. α_q and β_q are the b -bit value range and b is the number of bits after quantization. $x \in [\alpha, \beta]$ is the floating-point number before quantization in the range between α and β , $s = \frac{\beta - \alpha}{\beta_q - \alpha_q}$ is the scale and $z = \text{round}\left(\frac{\beta\alpha_q - \alpha\beta_q}{\beta - \alpha}\right)$ is the zero point. The output $x_q \in [\alpha_q, \beta_q]$ is the value after quantization. The same process is applied for weights, bias, and activation tensors.

$$x_q = \text{clip}\left(\text{round}\left(\frac{1}{s}x + z\right), \alpha_q, \beta_q\right) \quad (8)$$

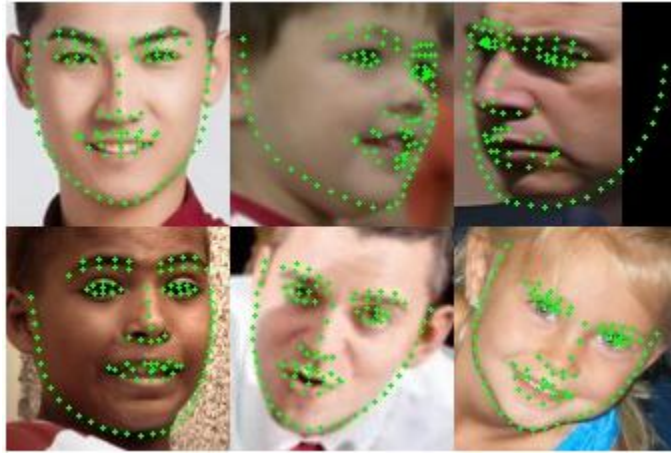
6.2 Technical Approach

Three types of quantization techniques are being used, and they are static quantization, dynamic quantization, and quantization awareness training. Both static quantization and dynamic quantization are post-quantization, meaning the model is quantized after training. Quantization awareness training means that the model is quantized during training. The difference between static quantization and dynamic quantization is what time the scale s and the zero point z of all the activation tensors are determined. s and z are calculated before model inference and they are fixed during inference. A dataset for calibration is used to calculate these two values. Dynamic quantization approach calculates these two values for activation tensors during reference and dynamic adjust them. The accuracy of dynamic quantization is slightly higher than static quantization.

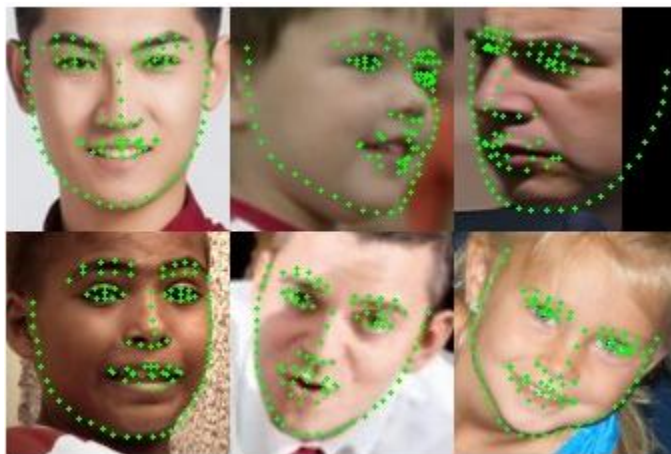
6.3 Experimental Results

In this research, static quantization using carefully selected calibration datasets is used. Different tools are available for static quantization such as TensorFlow Model Optimization Toolkit, PyTorch Quantization, ONNX Runtime and so on. The models in our research are trained by PyTorch and we intend to apply them in TI AM62A SoC for future research. To utilize TIDL platform in future research, ONNX Runtime is selected to quantize the models. The trained models are converted from PyTorch to ONNX. Then, ONNX Runtime is used to quantize the trained parameters in the ONNX models from floating point numbers to 8-bits integer values.

The experiment has finished. After model quantization, the size of neural network is reduced to one-fourth. The speed has increased by 97.8%. The accuracy is reduced by around 5%. Please see below examples in Figure 27 of facial landmark detect results before and after FLDNN model quantization. Visually, the facial landmarks detected almost the same before and after model quantization.



(a. Before model quantization)



(b. After model quantization)

Figure 27. Facial landmarks detect results before and after model quantization.

6.4 Summary

Based on the experiment result, it indicates that quantizing the deep learning neural network model is feasible with limited sacrifice of accuracy, but the size is reduced by four times, and the speed is almost double.

Each SoC has its own architecture design, advantages and hardware limitations. Different SoC may support different activity functions, neural network structures, loss functions and so on. So, further studies need to be explored in specific embedded environments.

CHAPTER SEVEN

SUMMARY

7.1 Overall Performance

To evaluate the overall performance of this learning-based driving monitoring system (DDMS), a video is taken in a vehicle in a real environment. Then this video is processed by DDMS to provide the driver ID, object detection results and driver status estimation results. Please see the example of the output from DDMS in Figure 28.

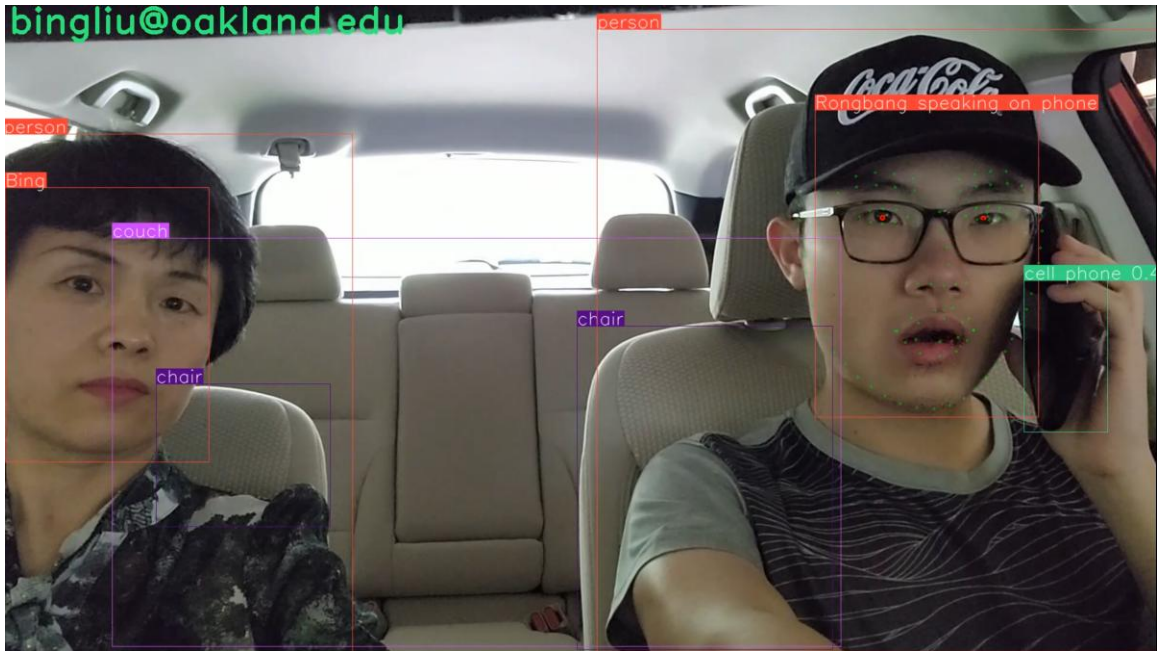


Figure 28. Output from DDMS.

The common objects such as humans, chairs and cellphones are detected. The camera is fixed so we can consider that the location of detected driver seat is fixed on the

images although the driver seat can be adjusted slightly. The person being detected in the location of driver seat is the driver. The face being detected in the driver's bounding box is the driver's face. The face image is sent to FRNN and FLDNN for driver ID identification and status estimation. The driver is recognized correctly along with the other person in the passenger seat by comparing the facial features produced by FRNN with pre-saved facial features in the system. The facial landmarks of the driver are nicely detected with glasses and a hat, and the gaze angle is calculated based on the landmarks' detection results. The calculation of gaze angles is explained below.

The facial landmarks of eyes are used to crop eyes' images. The eye's image is inverted then converted from color to gray image. A 3×3 kernel is used to erode and remove the noise. The threshold of (221,225) is applied to find the contours. The largest contour is used to calculate the minimum circle which completely covers the pupil with minimum area. The center of this minimum circle and the key points of the eye are used to calculate the gaze angle. For example, we can conclude that the driver looks towards to the left when the angle is smaller than 45 degrees horizontally and looks down when the angle is smaller than 45 degrees vertically. Similarly, the face angle can be calculated based on the relatively changed locations of the facial landmarks detected.

With detected cellphone near driver face, the movements of the driver's lips and the calculated gaze angle, the system concludes that the driver is speaking on the phone. The video is 85 seconds, and the driver status is correctly estimated for around 63 seconds. The overall accuracy of driver status estimation is around 75%. The names of the driver and the person on the passenger seat are 100% recognized correctly. The

common objects in the vehicle are detected correctly. The overall performance of the DDMS is effective.

Other status of drowsiness, looking around, texting, eating and talking can be estimated by our system as well. Please see below examples of detected texting status in Figure 29. The driver looks down and a cell phone is detected below the driver's face. Our system concludes that the driver is texting. This result indicates that our system works as expected in a dark environment as well. The original video before applying our DDMS is taken from <https://www.youtube.com/watch?v=c2YNKTUXL3w>.



Figure 29. The driver is texting.

7.2 Conclusion

In the research, a comprehensive study is conducted implementing a deep learning-based driving monitoring system (DDMS). The system can provide driver ID, estimate vehicle occupancy status and detect driver status using deep learning neural networks. The very wide and deep literary study conducted in this research proves that it is feasible to build such system.

A software, an object detection neural network (ODNN), a facial recognition neural network (FRNN) and a facial landmark detection neural network (FLDNN) are built in DDMS. The software drives the system by taking pictures using a camera installed on the dashboard of the vehicle and integrating these three deep learning models together to provide vehicle occupancy status, face ID identification results and driver status estimation results. ODNN provides object detection results such as the present status of human subjects, faces, dogs, and other common objects in the vehicle and the positions of the detected objects on the images. The driver's facial image is sent to FRNN and FLDNN. FRNN provides the driver ID. The facial landmarks detected by FLDNN and other objects detected by ODNN near the driver's faces are used to estimate the driver status such as activeness, drowsiness, distractedness and so on. Each model is evaluated quantitatively, and desired performance is achieved.

These three models are trained using floating points numbers so the parameters and the output from activation functions are floating points numbers. Quantization is used to convert the parameters in these models from floating point numbers to integer numbers to apply the models in the low-cost neural network inference environment.

To expand object detection training datasets, an Auto-Labeling Model (ALM) is trained in this research. The ALM expand the COCO dataset 2017 from 80 classes to 81 classes by adding human face class automatically. 4110 hours of human effort is saved.

7.3 Future Work

Further studies such as dynamic neural network quantization will be carried out to compare with static quantization. The performance difference between dynamic quantization and static quantization will be studied to choose the better strategy. Other studies such as integrating the DDMS into a production environment are also desired.

PUBLISHED PAPER DURING THIS RESEARCH: Received the best paper award

B. Liu and S. Ganesan, “Expand Object Detection Dataset Automatically via Deep Learning Neural Networks,” in *2023 3rd International Conference on Computing and Information Technology (ICCIT)*, 2023, pp. 421–426.

REFERENCES

- [1] “Toyota Enhances Pre-crash Safety System With Driver-monitoring Function.” [Online]. Available: <https://global.toyota/en/detail/248128>
- [2] “Backed by 20+ years of data-driven research.” [Online]. Available: <https://seeingmachines.com/technology/how-it-works/>
- [3] “BMW ConnectedDrive Driver Assistance systems in the BMW X5.” [Online]. Available: <https://www.bmw.ci/fr/tous-les-modeles/x-series/X5/2015/driverassistance.html>
- [4] H. Singh, J. S. Bhatia, and J. Kaur, “Eye tracking based driver fatigue monitoring and warning system,” in *India International Conference on Power Electronics 2010 (IICPE2010)*, Jan. 2011, pp. 1–6. doi: 10.1109/IICPE.2011.5728062.
- [5] Z. Guo, Z. J. Wang, and Z. Shen, “Physiological parameter monitoring of drivers based on video data and independent vector analysis,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 4374–4378. doi: 10.1109/ICASSP.2014.6854428.
- [6] A. Halin, J. G. Verly, and M. Van Droogenbroeck, “Survey and Synthesis of State of the Art in Driver Monitoring,” *Sensors*, vol. 21, no. 16, 2021, doi: 10.3390/s21165558.
- [7] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, “Object Detection in 20 Years: A Survey,” *Proceedings of the IEEE*, pp. 1–20, 2023.
- [8] Y. Wu, Y. Wang, S. Zhang, and H. Ogai, “Deep 3D Object Detection Networks Using LiDAR Data: A Review,” *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1152–1171, Jan. 2021.
- [9] A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya, “Object Detection Algorithms for Video Surveillance Applications,” in *2018 International Conference on Communication and Signal Processing (ICCSP)*, Apr. 2018, pp. 0563–0568.
- [10] P. K. Mishra and G. P. Saroha, “A study on video surveillance system for object detection and tracking,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2016, pp. 221–226.
- [11] K. Kalirajan and M. Sudha, “Moving Object Detection for Video Surveillance,” *The Scientific World Journal*, vol. 2015, p. 907469, Mar. 2015.
- [12] I. F. E. Babila, S. A. E. Villasor, and J. C. Dela Cruz, “Object Detection for Inventory Stock Counting Using YOLOv5,” in *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)*, May 2022, pp. 304–309.
- [13] A. Anjum, T. Abdullah, M. F. Tariq, Y. Baltaci, and N. Antonopoulos, “Video Stream Analysis in Clouds: An Object Detection and Classification Framework for High Performance Video Analytics,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1152–1167, Dec. 2019.
- [14] R.-Y. L. Lee, T.-Y. Chai, S.-Y. Chua, Y.-L. Lai, S. Y. Wai, and S.-C. Haw, “Cashierless Checkout Vision System for Smart Retail using Deep Learnin,” *Journal of System and Management Sciences*, vol. 12, no. 4, pp. 232–250, Apr. 2022.

- [15] M. Tan *et al.*, “Animal Detection and Classification from Camera Trap Images Using Different Mainstream Object Detection Architectures,” *Animals*, vol. 12, no. 15, 2022.
- [16] Y. Liu, Z. Ma, X. Liu, S. Ma, and K. Ren, “Privacy-Preserving Object Detection for Medical Images With Faster R-CNN,” *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 69–84, 2022.
- [17] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-View 3D Object Detection Network for Autonomous Driving.” arXiv, 2016. doi: 10.48550/ARXIV.1611.07759.
- [18] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” presented at the CVPR, San Diego, CA, USA: IEEE, Jun. 2005, pp. 886–893.
- [19] C. Zhou, L. Tang, S. Wang, and X. Ding, “Human Detection Based on Fusion of Histograms of Oriented Gradients and Main Partial Features,” in *2009 2nd International Congress on Image and Signal Processing*, Oct. 2009, pp. 1–5.
- [20] H. -X. Jia and Y. -J. Zhang, “Fast Human Detection by Boosting Histograms of Oriented Gradients,” in *Fourth International Conference on Image and Graphics (ICIG 2007)*, Aug. 2007, pp. 683–688.
- [21] N. G. Bardeh and M. Palhang, “New approach for human detection in images using histograms of oriented gradients,” in *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, May 2013, pp. 1–4.
- [22] X. Meng, J. Lin, and Y. Ding, “An extended HOG model: SCHOG for human hand detection,” in *2012 International Conference on Systems and Informatics (ICSAI2012)*, May 2012, pp. 2593–2596.
- [23] S. B. Thorat, S. K. Nayak, and J. P. Dandale, “Facial Recognition Technology: An analysis with scope in India.” 2010. [Online]. Available: <https://arxiv.org/abs/1005.4263>
- [24] T. Kanade, “Computer Recognition of Human Faces,” *Interdisciplinary Systems Research*, vol. 47, Jan. 1977.
- [25] L. Qing, S. Shan, and W. Gao, “Eigen-harmonics faces: face recognition under generic lighting,” in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, 2004, pp. 296–301. doi: 10.1109/AFGR.2004.1301547.
- [26] S. Z. Li *et al.*, “A near-infrared image based face recognition system,” in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, 2006, pp. 455–460. doi: 10.1109/FGR.2006.13.
- [27] A. Kumari Sirivarshitha, K. Sravani, K. S. Priya, and V. Bhavani, “An approach for Face Detection and Face Recognition using OpenCV and Face Recognition Libraries in Python,” in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2023, pp. 1274–1278. doi: 10.1109/ICACCS57279.2023.10113066.
- [28] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, Jan. 1991, doi: 10.1162/jocn.1991.3.1.71.
- [29] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust Face Recognition via Sparse Representation,” *IEEE Transactions on Pattern Analysis*

- and *Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009, doi: 10.1109/TPAMI.2008.79.
- [30] L. Zhang, M. Yang, and X. Feng, “Sparse representation or collaborative representation: Which helps face recognition,” *2011 International Conference on Computer Vision*, pp. 471–478, 2011.
 - [31] B. Moghaddam, W. Wahid, and A. Pentland, “Beyond eigenfaces: probabilistic matching for face recognition,” in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 30–35. doi: 10.1109/AFGR.1998.670921.
 - [32] W. Deng, J. Hu, J. Lu, and J. Guo, “Transform-Invariant PCA: A Unified Approach to Fully Automatic Face Alignment, Representation, and Recognition,” *IEEE Trans Pattern Anal Mach Intell*, vol. 36, no. 6, pp. 1275–1284, Jun. 2014, doi: 10.1109/TPAMI.2013.194.
 - [33] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, “Face recognition using Laplacianfaces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 328–340, 2005, doi: 10.1109/TPAMI.2005.55.
 - [34] C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan. 1998, pp. 555–562.
 - [35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
 - [36] G. G. Chrysos, E. Antonakos, P. Snape, A. Asthana, and S. Zafeiriou, “A Comprehensive Performance Evaluation of Deformable Face Tracking ‘In-the-Wild,’” *International Journal of Computer Vision*, vol. 126, no. 2–4, pp. 198–232, Feb. 2017, doi: 10.1007/s11263-017-0999-5.
 - [37] Y. Wu, T. Hassner, K. Kim, G. Medioni, and P. Natarajan, “Facial Landmark Detection with Tweaked Convolutional Neural Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 3067–3074, 2018, doi: 10.1109/TPAMI.2017.2787130.
 - [38] H. Zhang, Q. Li, Z. Sun, and Y. Liu, “Combining Data-Driven and Model-Driven Methods for Robust Facial Landmark Detection,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2409–2422, Oct. 2018, doi: 10.1109/tifs.2018.2800901.
 - [39] H. Lai *et al.*, “Deep Recurrent Regression for Facial Landmark Detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 5, pp. 1144–1157, 2018, doi: 10.1109/TCSVT.2016.2645723.
 - [40] D. Merget, M. Rock, and G. Rigoll, “Robust Facial Landmark Detection via a Fully-Convolutional Local-Global Context Network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.
 - [41] Y. Gizatdinova and V. Surakka, “Feature-based detection of facial landmarks from neutral and expressive facial images,” *IEEE Transactions on Pattern Analysis and*

- Machine Intelligence*, vol. 28, no. 1, pp. 135–139, 2006, doi: 10.1109/TPAMI.2006.10.
- [42] H. Song, J. Kang, and S. Lee, “ConcatNet: A Deep Architecture of Concatenation-Assisted Network for Dense Facial Landmark Alignment,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 2371–2375. doi: 10.1109/ICIP.2018.8451375.
 - [43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788.
 - [44] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755.
 - [45] Y. Xiong, K. Zhu, D. Lin, and X. Tang, “Recognize Complex Events from Static Images by Fusing Deep Channels,” in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, IEEE, 2015.
 - [46] S. Yang, P. Luo, C. C. Loy, and X. Tang, “WIDER FACE: A Face Detection Benchmark.” arXiv, 2015. doi: 10.48550/ARXIV.1511.06523.
 - [47] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection.” arXiv, 2017. doi: 10.48550/ARXIV.1708.02002.
 - [48] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 5962–5979, Oct. 2022, doi: 10.1109/tpami.2021.3087709.
 - [49] S. Chen, Y. Liu, X. Gao, and Z. Han, “MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices.” 2018.
 - [50] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” 2017.
 - [51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” 2015. [Online]. Available: <https://arxiv.org/abs/1512.03385>
 - [52] S. L, “Rigid-motion scattering for image classification.”
 - [53] L. Kaiser, A. N. Gomez, and F. Chollet, “Depthwise Separable Convolutions for Neural Machine Translation.” 2017. [Online]. Available: <https://arxiv.org/abs/1706.03059>
 - [54] K. KC, Z. Yin, M. Wu, and Z. Wu, “Depthwise separable convolution architectures for plant disease classification,” *Computers and Electronics in Agriculture*, vol. 165, p. 104948, 2019, doi: <https://doi.org/10.1016/j.compag.2019.104948>.
 - [55] Y. Dai, C. Li, X. Su, H. Liu, and J. Li, “Multi-Scale Depthwise Separable Convolution for Semantic Segmentation in Street–Road Scenes,” *Remote Sensing*, vol. 15, no. 10, 2023, doi: 10.3390/rs15102649.
 - [56] T. Zhang, X. Zhang, J. Shi, and S. Wei, “Depthwise Separable Convolution Neural Network for High-Speed SAR Ship Detection,” *Remote Sensing*, vol. 11, no. 21, 2019, doi: 10.3390/rs11212483.
 - [57] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “VGGFace2: A Dataset for Recognising Faces across Pose and Age,” in *2018 13th IEEE International*

- Conference on Automatic Face & Gesture Recognition (FG 2018)*, 2018, pp. 67–74. doi: 10.1109/FG.2018.00020.
- [58] K. Khabarлак and L. Koriashkina, “Fast Facial Landmark Detection and Applications: A Survey,” *Journal of Computer Science and Technology*, vol. 22, no. 1, p. e02, Apr. 2022, doi: 10.24215/16666038.22.e02.
 - [59] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, “Localizing Parts of Faces Using a Consensus of Exemplars,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2930–2940, Dec. 2013, doi: 10.1109/TPAMI.2013.23.
 - [60] X. Zhu and D. Ramanan, “Face detection, pose estimation, and landmark localization in the wild,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 2879–2886. doi: 10.1109/CVPR.2012.6248014.
 - [61] B. Luo, J. Shen, Y. Wang, and M. Pantic, “The iBUG Eye Segmentation Dataset,” in *2018 Imperial College Computing Student Workshop (ICCSW 2018)*, E. Pirovano and E. Graversen, Eds., in Open Access Series in Informatics (OASIs), vol. 66. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, p. 7:1-7:9. doi: 10.4230/OASIs.ICCSW.2018.7.