



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Doctor of Philosophy

**Driver Eye Status Monitoring System Based on
Lightweight Convolutional Neural Network
Architectures for Low-computing Devices**

The Graduate School

of the University of Ulsan

Department of Electrical, Electronic and Computer Engineering

Duy-Linh Nguyen

Driver Eye Status Monitoring System Based on Lightweight Convolutional
Neural Network Architectures for Low-computing Devices

Supervisor: Kang-Hyun Jo

A Dissertation

Submitted to
the Graduate School of the University of Ulsan
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Duy-Linh Nguyen

Department of Electrical, Electronic and Computer Engineering

University of Ulsan

June 2023

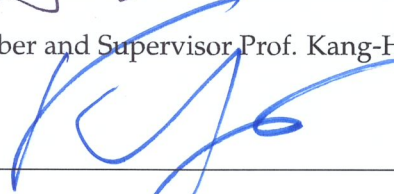
**Driver Eye Status Monitoring System Based on Lightweight
Convolutional Neural Network Architectures for Low-computing Devices**

This certifies that the dissertation
of Duy-Linh Nguyen is approved:

Committee Chair Prof. Hee-Jun Kang



Committee Member and Supervisor Prof. Kang-Hyun Jo



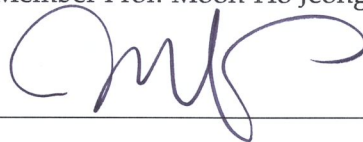
Committee Member Prof. Young-Soo Suh



Committee Member Prof. Hyun-Deok Kang



Committee Member Prof. Moon-Ho Jeong



Department of Electrical, Electronic and Computer Engineering

University of Ulsan, South Korea

June 2023

"Practice makes perfect."

Vietnamese Idiom

UNIVERSITY OF ULSAN

ABSTRACT

Graduate School of Electrical Engineering

Department of Electrical, Electronic and Computer Engineering

Doctor of Philosophy

Driver Eye Status Monitoring System Based on Lightweight Convolutional Neural Network Architectures for Low-computing Devices

by Duy-Linh Nguyen

Traffic accidents are the leading death rate among accident categories. One of the major causes of road traffic accidents is driver drowsiness. Many studies have paid attention to this issue and developed driver assistance tools to reduce the risk. These methods mainly analyze driver behavior, vehicle behavior, and driver physiology. This research proposes a driver eye status monitoring system based on lightweight convolutional neural network (CNN) architectures. The overall system consists of three stages: face detection, eye detection, and eye classification. In the first stage, the system utilizes a small real-time face detector based on the YOLOv5 network, named YOLO5Face (YOLO5nFace). The second stage focuses on exploiting the compact CNN network architecture combined with the inception network, and Convolutional Triplet Attention mechanism. Finally, the system uses a simple classification network architecture to classify open or closed eye status. Additionally, this work also provides the datasets for the eye detection task comprised of 10,659 images and 21,318 labels.

As a result, the real-time testing reached the best result at 33.12 FPS (frames per second) and 25.11 FPS on an *Intel*[®] *Core*[™] i7-4770 CPU @ 3.40GHz with 8 GB of RAM (Personal Computer - PC) and a 128-core Nvidia Maxwell GPU with 4 GB of RAM (Jetson Nano device), respectively. This speed is comparable with other previous techniques and it ensures that the proposed method can be applied in real-time systems for driver eye monitoring.

Acknowledgements

I would like to express my deep gratitude to my advisor, Professor Kang-Hyun Jo, who gave me the opportunity to study under his guidance, education, encouragement, and other support during my living and studying at the University of Ulsan. I would also thank the professors of the thesis committee: Prof. Hee-Jun Kang, Prof. Young-Soo Suh, Prof. Hyun-Deok Kang, and Prof. Moon-Ho Jeong, for their careful comments and valuable suggestions to improve the quality of my thesis.

Besides, I am grateful to all members of the Intelligent Systems Laboratory (IS-Lab) for the meaningful discussions and presentations. These lessons helped me to accumulate a lot of knowledge to pursue research in the Computer Vision field. Many thanks to the Brain Korea scholarship program (BK21), Department of Electrical, Electronic and Computer Engineering, University of Ulsan for allowing me to participate and present my knowledge at many scientific conferences around the world. Special thanks to my family, parents, wife, and daughter who always offered motivation to me to complete my duty.

Contents

Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 Motivation and Background	1
1.2 Problem Description and Objective	3
1.3 Contributions	3
1.4 Disposition	4
2 Literature Review	6
2.1 Computer Vision System	6
2.1.1 Image Classification	8
2.1.2 Object Detection	10
2.1.3 Attention Mechanisms	13
2.2 Traditional-based Methods	17
2.3 Machine learning-based Methods	18
3 Eye Detection Network	21
3.1 Network Architecture	21
3.1.1 Shrinking Module	21
3.1.2 Inception Module	23

3.1.3	Convolutional Triplet Attention Module	24
3.1.4	Detection Module	25
3.2	Loss Function	27
3.3	Experiments	29
3.3.1	Proposed Datasets for Eye Detection	29
3.3.2	Experimental Setup	31
3.3.3	Experimental Results	32
3.4	Ablation Studies	35
4	Eye Classification Network	37
4.1	Network Architecture	37
4.1.1	Backbone Module	38
4.1.2	Classification Module	39
4.2	Loss Function	40
4.3	Experiments	40
4.3.1	Datasets	40
4.3.2	Experimental Setup	40
4.3.3	Experimental Results	41
4.4	Ablation Studies	43
5	Real-time Eye Status Monitoring	45
5.1	YOLO5Face Detector	45
5.1.1	Network Architecture	47
5.1.2	Dataset	47
5.1.3	Experimental Results	48
5.2	Integrated System	48
5.3	Real-time Analysis	50

6 Conclusion	55
6.1 Conclusions	55
6.2 Future Works	57
A Publications	58
A.1 Journal	58
A.2 Conference	58
Bibliography	64

List of Figures

1.1	Overview of the proposed driver eye status monitoring system.	2
2.1	The simple human vision system (Elgendy, 2020).	7
2.2	The computer vision system for image classification (Elgendy, 2020).	7
2.3	Typical CNN architecture (Wikipedia contributors, 2023).	8
2.4	The VGG16 architecture for image classification.	9
2.5	Confusion matrix.	10
2.6	SSD architecture for object detection (Liu et al., 2016).	12
2.7	The visualization of IoU.	12
2.8	The SE architecture (Hu, Shen, and Sun, 2017).	15
2.9	The BAM architecture (Park et al., 2018).	16
2.10	The CBAM architecture (Woo et al., 2018).	17
3.1	The proposed eye detection network architecture.	21
3.2	An illustration of convolution layer.	22
3.3	The operation of the max pooling layer.	22
3.4	The architecture of the CReLU block.	23
3.5	The architecture of the inception layer.	23
3.6	The architecture of the CTA module.	24
3.7	The geometry of the predicted bounding box, anchor bounding box, and ground-truth bounding box.	26
3.8	NMS visualization in the eye detection.	28

3.9	The annotation generation process by Python code.	30
3.10	The interface of the LabelImg annotation tool on the FEI dataset.	30
3.11	The sample of annotation and XML file.	31
3.12	The proposed dataset distribution.	32
3.13	The qualitative results on five proposed datasets.	34
3.14	The several mistakes in detection results on five proposed datasets.	35
4.1	The proposed eye classification network architecture.	37
4.2	The operation of the average pooling layer.	38
4.3	The global average pooling layer.	39
4.4	The qualitative results of proposed eye classification network on CEW and MRL Eye datasets.	42
4.5	The confusion matrices on CEW and MRL Eye datasets.	43
5.1	The real-time testing setting with all devices.	45
5.2	The YOLO5Face architecture and sub modules.	46
5.3	The hardware is used in the real-time eye status monitoring system.	50
5.4	Block diagram of the proposed driver eye status monitoring system.	50
5.5	The qualitative results of the real-time eye status monitoring system with VGA video live-stream on Jetson Nano device with three partic- ipants.	51
5.6	The qualitative results of the real-time eye status monitoring system with VGA video live-stream on CPU-based PC with the single partic- ipant.	52
5.7	The qualitative results of the real-time eye status monitoring system with VGA video recording in the car on CPU-based PC.	52
5.8	The speed comparison between the proposed method and others in real-time testing on VGA (640 × 480) resolution.	54

List of Tables

3.1	The comparison result of eye detection network on individual proposed datasets.	33
3.2	Ablation studies of eye detection network on CEW dataset. The red color indicates the best result.	36
4.1	The comparison result of the eye classification network with popular classification networks on CEW and MRL Eye Datasets. The red color indicates the best competitors.	41
4.2	The comparison result of the eye classification network with different optimizers on CEW and MRL Eye Datasets.	43
4.3	The comparison result of eye classification networks with the GAP and fully connected (FC) layers on CEW and MRL Eye datasets.	44
5.1	The variants of YOLO5Face network (Qi et al., 2021).	48
5.2	Comparison of YOLO5Face networks and existing face detectors on the WIDER FACE validation dataset (Qi et al., 2021).	49
5.3	The speed performance of the system in real-time testing on PC and Jetson Nano device. The red color indicates the best result. (Unit of measurement: FPS).	53
5.4	The hardware specification of system in (Saurav et al., 2022) and proposed system.	53

List of Abbreviations

AI	Artificial Intelligence
AP	Average Precision
BAM	Bottleneck Attention Module
BioID	Biometric IDentification
BN	Batch Normalization
CBAM	Convolutional Block Attention Module
CEW	Closed Eyes in the Wild
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CReLU	Concatenated Rectified Linear Unit
CTA	Convolutional Triple Attention
DSFD	Dual Shot Face Detector
EAR	Eye Aspect Ratio
ECR	Eye Closure Ratio
FEI	Faculdade de Engenharia Industrial (Faculty of Industrial Engineering)
FERET	FacE REcognition Technology
FHD	Full High Definition
FPS	Frames Per Second
GAP	Global Average Pooling
GFLOP	Giga FLloating-point OPeration
GI4E	Gaze Interaction for (4) Everybody
GPS	Global Positioning System
GPU	Graphics Processing Unit
HD	High Definition
HOG	Histogram of Oriented Gradients
IoU	Intersection over Union

IPD	I nner P roduct D etector
KNN	K -Nearest Neighbor
ML	M achine L earning
MNIST	M odified N ational I nstitute of S tandards and T echnology
MRL	M edia R esearch L ab
NCS2	N eural C ompute S tick 2
NMS	N on- M aximum S uppression
PASCAL VOC	P ASCAL V isual O bject C lasses
PC	P ersonal C omputer
RAM	R andom A ccess M emory
RANSAC	R ANdom S AMple C onsensus
R-CNN	R egions with C onvolutional N eural N etwork
ReLU	R ectified L inear U nit
RoI	R egions of I nterest
SCRFD	S ample and C omputation R edistribution for E fficient F ace D etection
SE	S queeze-and- E xcitation
SIFT	S cale- I nvariant F eature T ransform
SSD	S ingle S hot D etector
SVM	S upport V ector M achine
VGA	V ideo G raphics A rray
VGG	V isual G eometry G roup
XML	E xtensible M arkup L anguage
YALEB	Y ALE F ace D ataset B
YOLO	Y ou O nly L ook O nce
ZJU	Z he J iang U niversity
RPi3	R aspberry P i 3
PAN	P ath A ggregation N etwork

List of Symbols

$[\cdot]$	Concatenation operation
\rightarrow	Transformation
\uparrow	Increase
\downarrow	Reducing
\in	Belongs to
Σ	Summation operation
\cap	Intersection operation
\cup	Union operation
\int	Integral operation
\otimes	Channel-wise multiplication operation
\cdot	Multiplication operation
$AP@\alpha$	Average Precision at α threshold
Avg	Average pooling
e^x	Standard exponential function
\mathcal{L}	Loss function
\log	Natural logarithm function
Max	Max pooling
\mathcal{P}	Probability
\mathbb{R}	Real numbers
\mathfrak{R}	Rotation operation
λ	Balancing parameter
$\mu_{\mathcal{M}}$	Mean of the minibatch \mathcal{M}
σ	Sigmoid function
$\sigma_{\mathcal{M}}$	Standard deviation of the minibatch \mathcal{M}
ϕ	Convolution operation

*Dedicated to
anyone who wants to study*

Chapter 1

Introduction

1.1 Motivation and Background

Road traffic accidents claim more than one million people every year and 90% are caused by driver distraction (*Road traffic injuries n.d.*). Driving distraction can occur at any time while the vehicle is operating due to many factors (Ansari, Naghdy, and Du, 2022). Drowsiness is a common cause of distracted driving. Therefore, it is necessary to build driver assistance systems to avoid accidents. A driver eye status monitoring system is an application built to alert drivers who are falling asleep. Drowsiness comes when the driver experiences a long journey, uses alcoholic drinks such as alcohol, beer or has a medical condition. Based on those observations, the researchers mainly focused on analyzing driver behavior, vehicle behavior, and driver physiology (Ramzan et al., 2019).

- Driver behavior can be recognized by extracting features of the driver body, like the features of eyes, mouth, hands, head posture, and body posture.
- Vehicle behavior is a method of surveillance for unusual vehicle movements when going out of a lane, swerving on the road, or interacting with other vehicles in the vicinity.
- Driver physiology is monitored through electrical pulses from the heart rate, blood pressure, and changes in body temperature.

The development of devices to monitor vehicle behavior and driver physiology require complex, high-cost devices and synchronous infrastructure. Moreover, wearable devices can cause discomfort to the driver while operating and several natural

factors in the driver body can interfere with the received electrical signal.

Realizing the importance of eye status in the early stage (Valenti and Gevers, 2011) of sleep and the above analysis, this research proposes a driver eye status monitoring system. This system is based on lightweight convolutional neural networks (CNNs) with a new version of attention mechanisms, named the Convolutional Triplet Attention module. The use of lightweight CNN architectures to optimize network parameters for low-computing devices while it is still ensuring the extraction of useful facial and eye information for processing. On the other hand, the attention mechanism helps the detector to focus on processing information in the eye area and ignore useless or background information. The proposed system aims to detect the status of the driver eye with three stages: face detection, eye detection, and eye classification. The proposed system works as the mechanism of the driver eye status monitoring system through the front-mounted camera. It does not influence the operation or does not cause unpleasant effects on the driver's body, like the above-mentioned methods. Figure 1.1 shows the overview of the proposed driver eye status monitoring system.

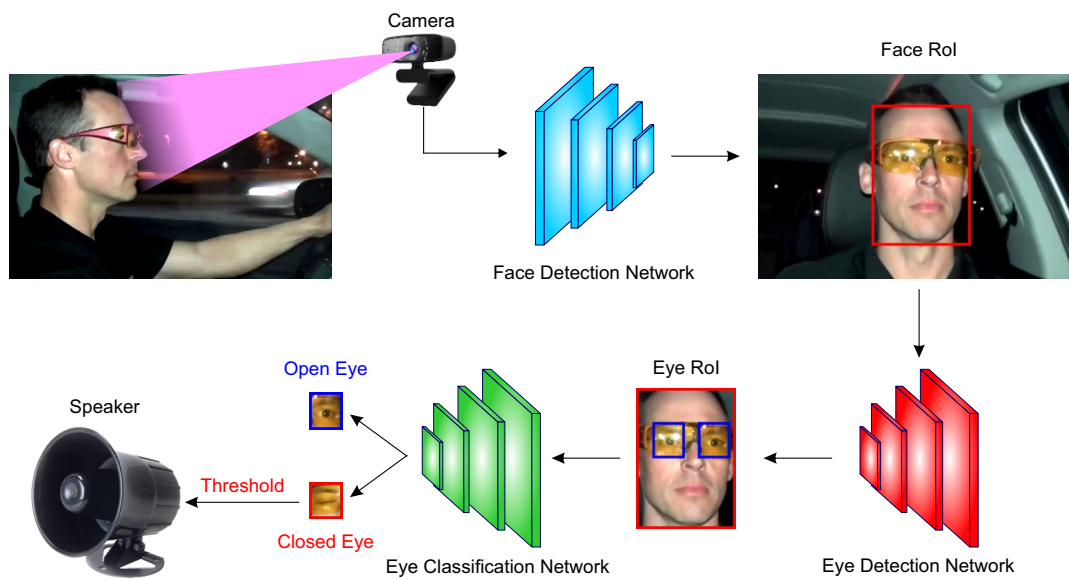


FIGURE 1.1: Overview of the proposed driver eye status monitoring system.

1.2 Problem Description and Objective

The research related to human eye monitoring systems is mainly based on two approaches, including traditional-based image processing and machine learning-based methods. In general, traditional methods apply computationally complex techniques and are very sensitive to environmental lighting conditions and facial occlusion, although they provide useful information.

Besides, machine learning methods focus on exploiting the features of the eyes and facial organs related to the sleepy state such as the eyelids and mouth. Meanwhile, these organs are quite small in the image and lack specialized datasets for detection tasks. It requires complex techniques, high computational costs to accurately detect those locations, and high labor costs to annotate the datasets.

Therefore, in order to simplify eye position detection and reduce the computational cost of the system, this research proposes an effective method of eye location detection and eye classification based on lightweight CNN architectures. Additionally, this work has also provided the datasets for eye position detection which is popularly used in the object detection field.

1.3 Contributions

This work evaluates and exploits the advantages of previous works to develop a real-time driver eye monitoring system. The results are inspected based on the performance of each proposed network on the respective datasets. Then the integrated system is tested in real-time and the recorded video. The main contributions of this research are shown as follows:

- This work proposes two lightweight CNNs for eye detection and eye classification tasks. These networks aggregated with a real-time face detector for mobile and edge devices to build a three-stage driver eye status monitoring system. The CNNs in this paper are built for use in low-computing devices like a CPU and an Nvidia Jetson Nano. With network parameter optimization of compact architectural designs, the proposed networks solve the problem of computational and deployment costs. In addition, it is not invasive to the driver during

use.

- It also provides the datasets for locating the eye area in images under different situations and follows the PASCAL VOC dataset format. These are basic datasets for machine learning developers who use features from the driver eye. Along with the proposed eye detection and eye classification networks, they can use in other fields such as eye-tracking, medical, and biomedical.
- On the application side: the proposed system is tested in real-time on a CPU-based personal computer and a Jetson Nano device without high latency while ensuring accuracy.

1.4 Disposition

This part presents the organization of the manuscript:

Chapter 2 is a general introduction to a computer vision system, convolutional neural network (CNN), image classification, object detection, and attention mechanisms used in CNN. In addition, it discusses the traditional and current methods related to eye classification, eye detection, and eye monitoring.

Chapter 3 describes the proposed eye detection network architecture and dataset. The eye detection network is designed using a combination of shrinking, inception, Convolutional Triplet Attention, and detection modules. The proposed dataset for the eye detection task includes 10,659 images and 21,318 annotations for eye positions following the PASCAL VOC dataset format.

Chapter 4 explains the proposed eye classification network architecture. This network is built on the basic layers of the common classification networks and improves it by using the global average pooling layer replace to the fully connected layers. This technique is intended to simplify the network architecture and save a lot of network parameters.

Chapter 5 discusses and analyzes the proposed real-time driver eye status monitoring system on low-computing devices. It includes the hardware configuration, the speed test results, visualization results in the laboratory environment and various driving scenarios videos.

Chapter 6 concludes the issue and summarizes directions for future work.

Chapter 2

Literature Review

This study focuses on three main issues, including eye detection, eye classification, and attention mechanisms to build lightweight CNN networks for real-time systems. This chapter will cover the research related to eye detection, eye classification, and human eye status monitoring systems.

Chapter 2 is divided into three important parts. The first part introduces an overview of computer vision and several tasks. The second and third sections present the traditional methods and machine learning methods related to human eye classification, detection, and monitoring.

2.1 Computer Vision System

The core issue of artificial intelligence (AI) systems is to perceive the world around them and perform actions automatically using the analysis of perceptions. Visual perception is the science of perceiving and understanding the world through images and videos by building a physical model on which an AI system can take appropriate actions (Elgendy, 2020).

Human vision is a system that works based on the mechanisms of the human body and other living organisms. It uses the senses or eyes to capture images and then the brain processes this information to help it understand. A simple human vision system is shown in Figure 2.1

Similarly, computer vision allows AI machines can train themselves to perform

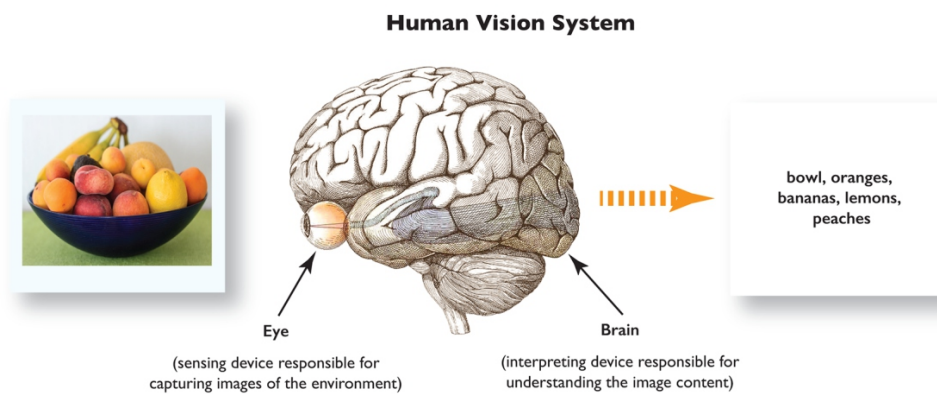


FIGURE 2.1: The simple human vision system (Elgandy, 2020).

the above processes. Such a machine system uses a combination of sensors (cameras), algorithms, and data sets to perform specific tasks. Figure 2.2 depicts a computer vision system for the image classification task.

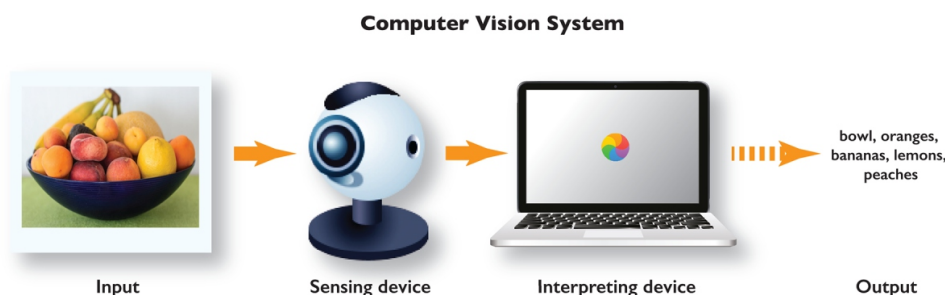


FIGURE 2.2: The computer vision system for image classification (Elgandy, 2020).

Computer vision consists of two key technologies, including machine learning and convolutional neural networks. Machine learning (ML) uses algorithm-based models to drive computers to understand context through visual data analysis. The machine applies these algorithms to automatically learn and solve the issue in the most suitable way. The convolutional neural network considers the image by subdividing it into pixels. Each pixel is assigned a label or tag. These labels are then collectively used to perform the convolution operation. Via this process, convolutional neural networks can process visual inputs and extract feature maps for various tasks. Figure 2.3 is a typical convolutional neural network.

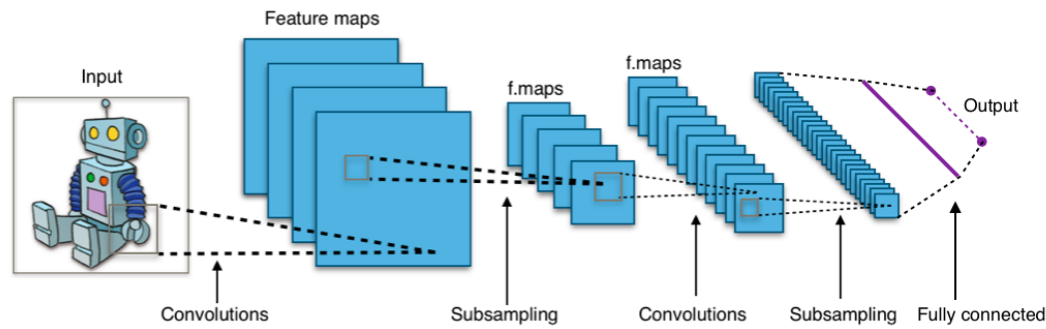


FIGURE 2.3: Typical CNN architecture (Wikipedia contributors, 2023).

2.1.1 Image Classification

Image classification is an important task in the Computer Vision field, which is used to classify images into different groups based on their content (Wang and Su, 2019). The goal of image classification is to help computers automatically recognize and classify images accurately and quickly, making Computer Vision applications smarter. Image classification has many practical applications such as facial recognition, product classification, medical image analysis, etc. The basic steps in image classification are:

- (1) **Preprocessing:** noise reduction and normalizing image dataset for the training model.
- (2) **Modeling:** design a suitable machine learning model and train on the dataset to find the special features of the images and classify them.
- (3) **Model evaluation:** evaluate the accuracy of the model on the test set, fine-tune, and improve the model.

Common methods for performing image classification include traditional-based and CNN-based methods. Traditional-based methods are used for image classification before deep learning models appeared such as K-Nearest Neighbor (KNN) (Mucherino, Papajorgji, and Pardalos, 2009), Decision Tree (Fürnkranz, 2010), and Support Vector Machine (SVM) (Cortes and Vapnik, 1995). KNN is a simple method for image classification. It is operated on the distance between the classified image and the images in the training set. The images in the training set are sorted by distance to the image to be classified and the results are the number of images belonging to each nearest class. The Decision Tree is built by dividing the dataset into subsets

with their attributes. When the image to be classified is passed through the Decision Tree, it is classified according to the class corresponding to the leaf node. SVM finds the best hyperplane to separate classes. When the image goes through the SVM, it is classified by the layer located on the other side of the hyperplane.

Convolutional Neural Network is one of the most popular techniques in the image classification task. CNN is a type of artificial neural network designed to extract features from images and use them for image classification. CNNs are designed based on convolution layers, pooling layers, and fully connected layers. Convolutional layers are used to extract local features of an image, where filters are applied to search for features such as edges, shapes, and colors. Pooling layers are used to reduce the size of the image and help reduce the complexity of the model. After extracting the features, the fully connected classes are used to classify the images. These fully connected layers transform the extracted feature map into class predictions of the image. To train a CNN, the labeled dataset is provided, consisting of images and corresponding labels for each image. Then, the network applies the optimization algorithms to adjust the weights of the model that can properly classify the images in the training set. AlexNet (Krizhevsky, Sutskever, and Hinton, 2017), VGG (Simonyan and Zisserman, 2014), GoogleNet (Szegedy et al., 2015), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), EfficientNet (Tan and Le, 2019), and MobileNet (Howard et al., 2017) architectures achieve good performance in image classification on datasets like ImageNet, CIFAR-10, CIFAR-100, and MNIST. Figure 2.4 shows the representative architecture of the image classification network (VGG16).

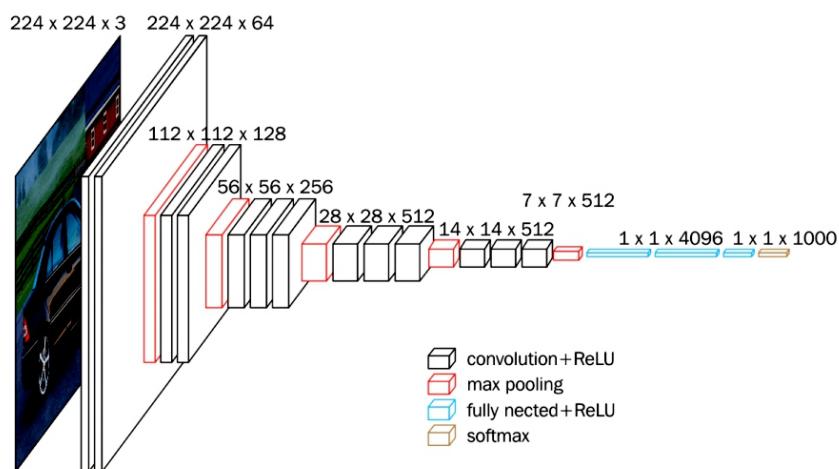


FIGURE 2.4: The VGG16 architecture for image classification.

To evaluate the performance of the classification network, many different metrics can be used.

Accuracy: The most widely used metric is accuracy and it is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

True positive (TP): A test result that correctly indicates the presence of a condition or characteristic.

True negative (TN): A test result correctly indicates the absence of a condition or characteristic.

False positive (FP): A test result wrongly indicates that a particular condition or attribute is present.

False negative (FN): A test result wrongly indicates that a particular condition or attribute is absent.

Confusion Matrix: Confusion matrix is a table that presents the number of predicted and actual class labels of the classification algorithm. This table provides a detailed analysis of the network performance of each class as shown in Figure 2.5.

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

FIGURE 2.5: Confusion matrix.

2.1.2 Object Detection

Object detection is an important and challenging task in Computer Vision research on the recognition of objects in digital images or videos. The goal of object detection is to find and draw a bounding box that determines the position and size of objects in the image and assigns each bounding box a label corresponding to the type of object.

Object detection is widely applied in various fields such as autonomous vehicles, surveillance systems, facial detection, industrial automation, etc. Object detection has generally gone through two historical periods corresponding to two techniques: traditional-based object detection and CNN-based detection (Zou et al., 2023).

Traditional-based methods are usually based on identifying local features in images and then applying classifiers to classify objects following these features. Haar Cascades (Viola and Jones, 2001) are commonly used in face detection, a method of finding common features in images, such as edges and corners. Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) is an object detection method based on HOG features, which is a representation of directional features in an image. Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) uses a classifier to detect local features in the image, then applies a SIFT algorithm to identify those features. Then, the objects are classified by comparing these SIFT features with a training set.

CNN-based methods are divided into two main groups: single-stage detection and two-stage detection. Single-stage detection is designed to detect and locate objects in an image with only one pass through the neural network, rather than going with multiple stages. This increases processing speed and reduces computation costs, especially in real-time applications. Among the typical single-stage detectors are You Only Look Once (YOLO) (Redmon et al., 2016) and its variants, Single Shot Detector (SSD) (Liu et al., 2016), RetinaNet (Lin et al., 2017), and EfficientDet (Tan, Pang, and Le, 2019). In contrast, two-stage object detectors are developed to search for regions of interest (RoI) in images before object detection. Popular two-stage object detectors are the Regions with Convolutional Neural Network (R-CNN) (Girshick et al., 2013), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), and Mask R-CNN (He et al., 2017). Two-stage detectors generally offer high detection accuracy but computational complexity and speed are lower than single-stage detectors. Figure 2.6 shows the representative architecture of a single-stage detector (SSD).

Each object detection network will use different measurement units depending on each specific purpose. Several standard evaluation metrics used in object detection algorithms include:

Intersection over Union (IoU): IoU measures the overlap between the predicted bounding boxes and the ground truth bounding boxes. It is calculated by dividing

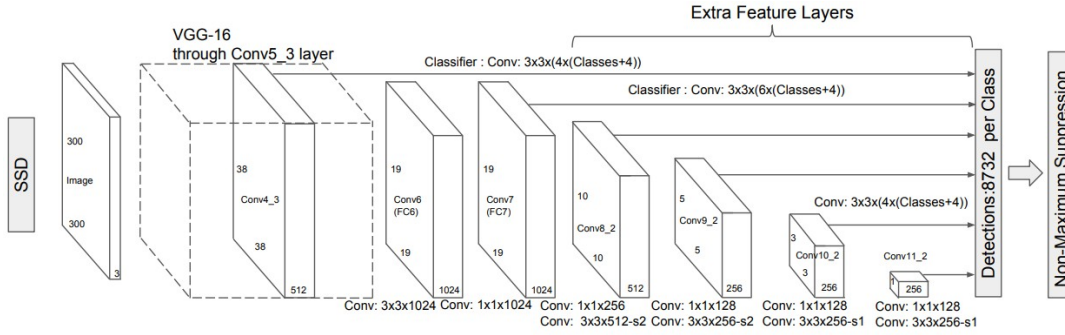


FIGURE 2.6: SSD architecture for object detection (Liu et al., 2016).

the intersection area between the two boxes by the area of their union. The Equation of IoU is defined as:

$$IoU = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}}, \quad (2.2)$$

where, B_p and B_{gt} are the predicted bounding box and ground-truth bounding box, respectively. \cap is the intersection operation. \cup is the union operation. Figure 2.7 shows a visual representation of the IoU.

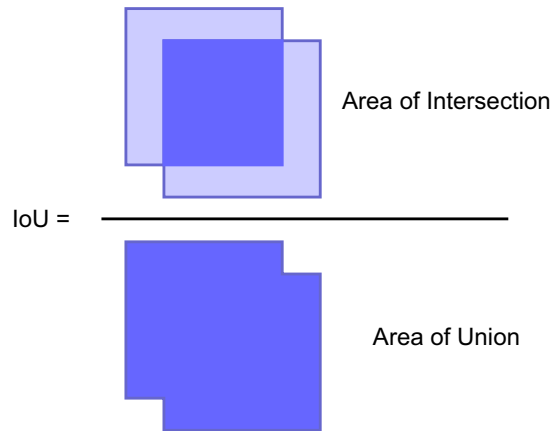


FIGURE 2.7: The visualization of IoU.

Recall: The recall is the fraction of true positive detections over the sum of true positive and false negative samples. The Equation of recall is defined as:

$$Recall = \frac{TP}{TP + FN}. \quad (2.3)$$

Precision: The precision is the fraction of true positive detections over the sum of true positive and false positive samples. The Equation of precision is defined as:

$$Precision = \frac{TP}{TP + FP}. \quad (2.4)$$

Average Precision (AP): The average precision is a commonly used metric to evaluate object detection performance. It is calculated as the area under the precision-recall curve. The precision-recall curve is generated by varying the detection threshold and plotting precision against recall at each threshold value. The Equation of AP is defined as:

$$AP@{\alpha} = \int_0^1 p(r)dr, \quad (2.5)$$

where $AP@{\alpha}$ means the AP is evaluated at IoU threshold α (e.g.: AP@50 at $\alpha = 0.5$, AP@75 at $\alpha = 0.75$, AP@0.5:0.95 at ten α thresholds ($\alpha = 0.5$ to 0.95 with step is 0.05)).

Mean Average Precision (mAP): The mean average precision is the mean of the AP values across all different object classes in the dataset. This is a popular metric for evaluating multi-class object detection algorithms. The Equation of mAP is defined as:

$$mAP = \frac{1}{c} \sum_i^c AP_i, \quad (2.6)$$

where c is the number of classes and AP_i is AP of class i -th.

F1 score: F1 score is the harmonic mean of precision and recall. It is a measure of the balance between precision and recall. The Equation of F1 score is defined as:

$$F1 \text{ score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.7)$$

2.1.3 Attention Mechanisms

The attention module is a necessary component in convolutional neural networks, which are used to improve feature extraction and model efficiency. In traditional CNN, each convolution layer applies a filter (kernel) to extract features from the input. However, not all features are important and should be focused on. Therefore, an attention mechanism is utilized to weighting the features and select only the most important ones. The attention mechanism works by calculating the weights of each feature. These weights are used to compute the weighted average of those features.

This result is a new input for the next layer. In this manuscript, the attention mechanism is considered on two groups, spatial attention and channel attention. Spatial attention is split into two subgroups, local attention and global attention. Some representatives of these concentration mechanisms are Squeeze-and-Excitation Network (SE) (Hu, Shen, and Sun, 2017), Bottleneck Attention Module (BAM) (Park et al., 2018), Convolutional Block Attention Module (CBAM) (Woo et al., 2018), and Convolutional Triple Attention (CTA) (Misra et al., 2020).

Squeeze-and-Excitation: SE uses a channel attention mechanism to focus on the input channels, which helps the model select the important channels and discard the unhelpful ones. SE is implemented through two stages: squeeze and excitation. During the squeeze phase, SE applies a global average pooling layer to reduce the size of the feature map to a vector. Then, in the excitation phase, SE utilizes a multilayer neural network to compute the weights for the input channels. These weights are applied to the original input feature map by element-wise multiplication to get an output feature map with attention weights. The SE attention module is composed of three operations: squeeze (F_{sq}), excitation (F_{ex}), and scaling (F_{scale}). The SE attention mechanism is shown as follows:

$$X \in \mathbb{R}^{H' \times W' \times C'} \xrightarrow{F_{tr}} U \in \mathbb{R}^{H \times W \times C} \quad (2.8)$$

$$x_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j), \quad (2.9)$$

$$y_c = F_{ex}(x, w) = \sigma(w_2 ReLU(w_1 x)), \quad (2.10)$$

$$z_c = F_{scale}(u_c, y_c) = u_c \otimes y_c, \quad (2.11)$$

where X is the input feature map. F_{tr} is transformation as a convolution operation. u_c is c -th channel of the feature map U . σ is the sigmoid activation function. $ReLU$ is the rectified linear activation function. $w_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, $w_2 \in \mathbb{R}^{C \times \frac{C}{r}}$, and $x = [x_1, x_2, \dots, x_c]$. \otimes denotes the channel-wise multiplication operation. Figure 2.8 shows the architecture of SE.

Bottleneck Attention Module: The mechanism of BAM is implemented through

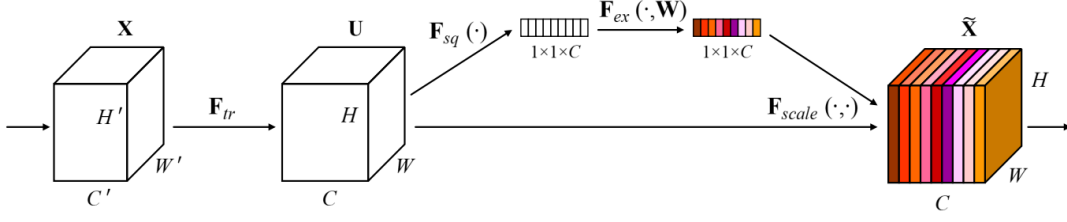


FIGURE 2.8: The SE architecture (Hu, Shen, and Sun, 2017).

two stages: channel attention and spatial attention. During the channel attention phase, the BAM handles a structure similar to SE to compute the weights for the input channels. Then, in the spatial attention phase, BAM utilizes a convolutional neural network to compute the weights for the locations on the input feature map. These weights are used to perform element-wise multiplication and element-wise addition to concentrate on the more important positions, minimizing the influence of the unimportant positions on the prediction results. The BAM attention mechanism is shown as follows:

$$F' = F + F \otimes M(F), \quad (2.12)$$

$$M(F) = \sigma(M_c(F) + M_s(F)), \quad (2.13)$$

$$M_c(F) = BN(MLP(Avg(F))), \quad (2.14)$$

$$M_s(F) = BN(\phi_3^{1 \times 1}(\phi_2^{3 \times 3}(\phi_1^{3 \times 3}(\phi_0^{1 \times 1}(F))))), \quad (2.15)$$

where $F' \in \mathbb{R}^{H \times W \times C}$ is the refined feature map. $F \in \mathbb{R}^{H \times W \times C}$ is the input feature map. $M(F) \in \mathbb{R}^{H \times W \times C}$ is an inference feature map. $M_c(F) \in \mathbb{R}^C$ is the channel attention. $M_s(F) \in \mathbb{R}^{H \times W}$ is the spatial attention. σ is a sigmoid function. BN is a batch normalization operation. MLP is a multi-layer perception. ϕ denotes a convolution operation. Figure 2.9 shows the structure of BAM.

Convolutional Block Attention Module: CBAM also includes two main parts: channel attention and spatial attention. However, unlike SE and CBAM, this mechanism uses a sequence of the above two components. The channel attention helps to attend to the important characteristics of each input channel and provides a channel

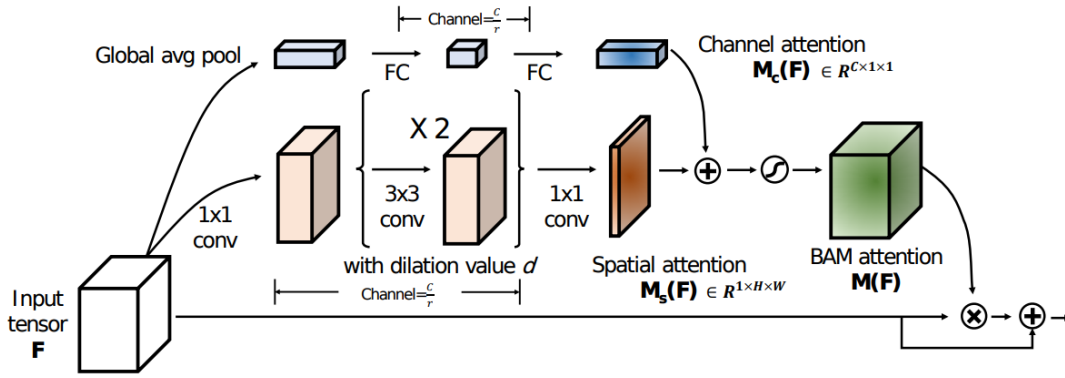


FIGURE 2.9: The BAM architecture (Park et al., 2018).

weight map. Spatial attention concerns the important locations of the image and gives a spatial weight map. These two parts are combined to create a comprehensive attention map for the entire input. The CBAM attention mechanism is shown as follows:

$$\begin{aligned} F^1 &= M_c(F) \otimes F, \\ F^2 &= M_s(F^1) \otimes F^1, \end{aligned} \quad (2.16)$$

in which, the symbol \otimes represents the element-wise multiplication operation. F^1 and F^2 present the intermediate and output feature maps, respectively. M_c and M_s are computed by the following equations:

$$\begin{aligned} M_c(F) &= \sigma(MLP(\mathcal{A}vg(F)) + MLP(\mathcal{M}ax(F))), \\ M_s(F) &= \sigma(\phi^{7 \times 7}([\mathcal{A}vg(F), \mathcal{M}ax(F)])), \end{aligned} \quad (2.17)$$

where σ describes the sigmoid function. $[\cdot]$ symbol denotes the concatenation operation. MLP , $\mathcal{A}vg$, $\mathcal{M}ax$, $\phi^{7 \times 7}$ are the multilayer perception, the average pooling layer, the max pooling layer, and 7×7 standard convolution layer, respectively. Figure 2.10 shows the structure of CBAM.

Convolutional Triple Attention: CTA is designed to solve the problem of calculation independence on the channels and space of the input data. CTA combines three different attention techniques, including channel attention, spatial attention, and relational attention using rotation operations. Channel attention and spatial attention help focus on important features of channels and locations over the entire input image data, respectively. Relational attention helps find interactions between

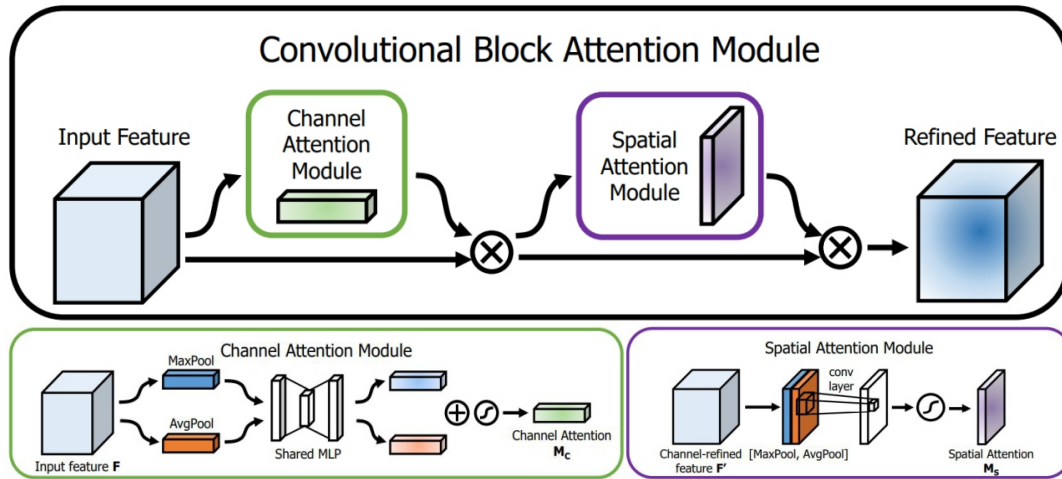


FIGURE 2.10: The CBAM architecture (Woo et al., 2018).

features of an image, creating a relationship map to improve the model's representation. More details of Convolution Triple Attention with application case will be explained in Chapter 3.

2.2 Traditional-based Methods

Traditional techniques rely heavily on the geometrical structure of the eye to detect the center eye position in an image or video.

The work in (Valenti and Gevers, 2011) applied the Isophotes Curvature Estimation technique to determine the center of the eye and then builds a voting system for that point. This method was tested on the BioID and the color FERET datasets. It achieved a significant improvement in accuracy and robustness over state-of-the-art techniques for eye center location in standard low-resolution imagery.

The authors in (Markuš et al., 2014) introduced a method using an ensemble of randomized regression trees to locate the pupil of the eye. The implementation achieved quite good real-time performance on mobile devices. The proposed software provided the solution for a low-cost replacement to complex, high-cost, and commercial eye-tracking systems.

The method in (Timm and Barth, 2011) localized the center of the eye based on the image gradient and applied a squared dot product between the center candidates and the image gradient. The experiment was comprehensively evaluated on the

BioID challenge dataset. The implementation method was quite simple but highly effective in both accuracy and robustness.

The research in (Świrski, Bulling, and Dodgson, 2012) used the Haar-like feature combined with a K-means cluster to locate the pupil and fit an ellipse to the pupil using the novel image-aware Random Sample Consensus (RANSAC) technique. This work compared the proposed approach against the existing real-time pupil tracking system using the manually labeled infrared dark-pupil eye dataset. This technique had a higher pupil detection rate and greater pupil tracking accuracy than other competitors.

The approach in (Araujo et al., 2014) proposed an Inner Product Detector (IPD) based on correlation filters to detect pupil landmarks. The main contributions of the method were the tolerance to small variances in the desired patterns, a low computational cost, and the generalization of other features. The experiment was evaluated on the BioID dataset and compared to state-of-the-art techniques.

2.3 Machine learning-based Methods

The eye is an important organ of the human body and its status could reflect the first stage of drowsiness. From that observation, many machine learning studies have achieved high efficiency by focusing on exploiting and monitoring human eye location. Several basic machine learning methods have been applied, such as the Haar wavelet and Support Vector Machine (SVM) (Chen and Liu, 2015), Histogram of Oriented Gradient-based Support Vector Machine (HOG-SVM) (Sharma and Savakis, 2015), self-similarity information combined with shape analysis (Leo et al., 2013), and Viola-Jones (Chirra, Reddy, and KishoreKolli, 2019). These techniques could overcome the disadvantages of traditional methods and achieve higher accuracy. However, they also need to be used in conjunction with other modern methods for implementation in real-time systems.

Nowadays, the impressive advances of machine learning have made the development of computer vision applications based on CNNs more and more popular. In this trend, eye and pupil detection also used depth and complicated CNNs for improved extraction features and learning them. These CNNs were combined with

multiple tasks to fully exploit the ability to learn important information from feature maps.

The experiment in (Khan et al., 2019) used the eyelid curvature angle as the basis for determining whether the eye is closed or open. The network was evaluated on three datasets. In the first dataset with uniform background, the proposed method achieved an accuracy of 95%. For another benchmark dataset with significant variations based on face deformations, it reached an accuracy of 70%. Finally, this method got an accuracy of 95% on the real-time video dataset. From that, the proposed network was feasible for use in real-time systems.

The research in (Shakeel et al., 2019) considered drowsiness detection as an object detection task by identifying and detecting closed or open eyes. This work used a combination of MobileNet with Single Shot Multibox Detector (SSD) as an object detection task. A custom dataset was selected from 6,000 images labeled with face, open eye, and closed eye. Of these, 350 images were randomly selected for the testing phase. This experiment achieved 84% of mean average precision (mAP). In addition, it could be applied to embedded and mobile devices in vehicles.

The approach in (Sathasivam et al., 2020) used the facial landmarks technique to calculate the Eye Aspect Ratio (EAR) and Eye Closure Ratio (ECR) for drowsiness detection. A real-time system was also developed with the Pi camera, Raspberry Pi 4, and Global Positioning System (GPS) to detect and analyze continuously the eye statuses. This system could recognize whether the driver is drowsy or not with different sleeping levels. The experiment was conducted successfully and gave 90% of accuracy.

The study in (Saurav et al., 2022) proposed a vision-based system for real-time eye state recognition on an embedded platform. The system used an ensemble of two lightweight convolutional neural networks, each being trained to extract relevant information from the eye areas. The experiment was evaluated on three datasets: ZJU, CEW, and MRL. It achieved the best accuracy of 97.99% on the ZJU dataset and competitive accuracy to others on the remaining datasets. Besides, this work reported the speed testing results on the Nvidia Xavier device and the Intel NCS2 embedded platform with 62 FPS and 11 FPS, respectively.

Another work (Jahan et al., 2023) introduced a 4D model for eye categorization

by improving the VGG architecture family. This network was trained and evaluated on the MRL Eye dataset. As a result, the 4D network reached an accuracy of 97.53% on the test set and outperformed other VGG network versions (VGG16, VGG19). This work also provided a solution to implement a complete drowsiness detection system.

In summary, machine learning methods exhibit outstanding advantages in feature extraction and eye location detection. In general, these methods focus on exploiting the features of the eyes and facial organs related to the sleepy state such as the eyelids and mouth. Meanwhile, these organs are quite small in the image and lack specialized datasets for detection tasks. It requires complex techniques, high computational costs to accurately detect those locations, and high labor costs to annotate the datasets. Therefore, to simplify eye position detection and reduce the computational cost of the system, this research proposes an effective method of eye location detection and eye classification based on lightweight CNN architectures. Besides, this work has also provided the datasets for eye position detection which is popularly used in the object detection field.

Chapter 3

Eye Detection Network

An eye detection network is used in the second stage of the eye status detection system. The detailed architecture description of this network is shown in Figure 3.1. It is built with four following modules, named the shrinking, inception, Convolutional Triplet Attention, and detection modules.

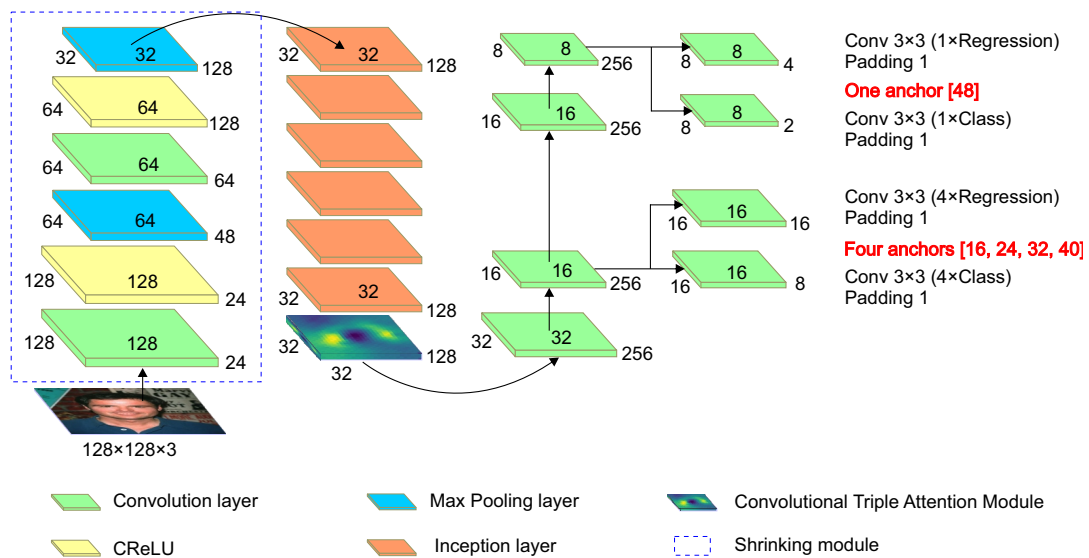


FIGURE 3.1: The proposed eye detection network architecture.

3.1 Network Architecture

3.1.1 Shrinking Module

This module mainly uses convolution layers with a kernel size of 3×3 and strides of 1, 2, 1, 2 in the convolution and max pooling layers, respectively. It helps to extract the basic features of the eye and reduce the input image from 128×128 to 32×32 .

The convolution operation involves taking a small matrix called a kernel or filter, sliding it over an input image or feature map, and performing element-wise multiplication between the kernel and the corresponding input image region. The resulting values are then summed up to produce a single output value in the output feature map. An example of convolution operation is shown in Figure 3.2.

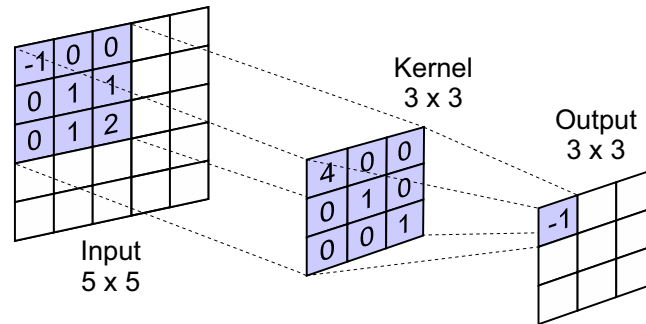


FIGURE 3.2: An illustration of convolution layer.

The max pooling layer is a down-sampling operation that reduces the dimensionality of the feature maps and makes the network more computationally efficient. This layer works by dividing the input image into a set of non-overlapping rectangular regions (pooling windows) and outputs the maximum value within each window. The operation of the max pooling layer can be illustrated in Figure 3.3.

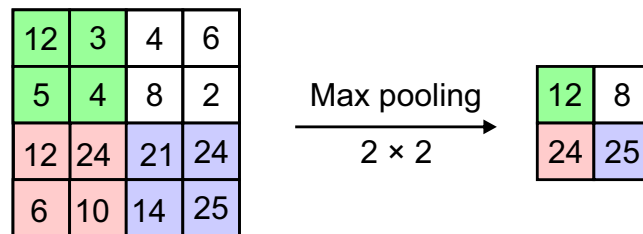


FIGURE 3.3: The operation of the max pooling layer.

In addition, the Concatenated Rectified Linear Unit (CReLU) module (Shang et al., 2016) is also used to increase the efficiency of feature extraction. The CReLU is a variation of the Rectified Linear Unit (ReLU) activation function. For a traditional ReLU function, the output is equal to the input if it is greater than zero, and zero otherwise. The ReLU activation function is defined as:

$$\text{ReLU} = \max\{0, x\}. \quad (3.1)$$

On the other hand, CReLU doubles the number of feature maps by concatenating

the ReLU activation with its negation along the channel dimension. The CReLU activation function can be expressed as:

$$CReLU(x) = [ReLU(x), ReLU(-x)], \quad (3.2)$$

where $[\cdot]$ denotes concatenation along the channel dimension. The architecture of CReLU is described as shown in Figure 3.4.

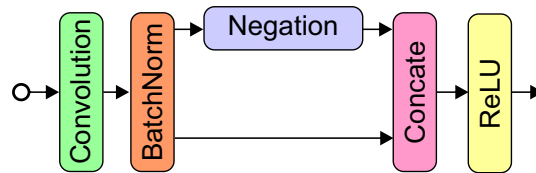


FIGURE 3.4: The architecture of the CReLU block.

3.1.2 Inception Module

The inception module is made up of six inception layers (Szegedy et al., 2014). Each inception layer contains four convolution branches. Figure 3.5 shows the structure of the inception layer.

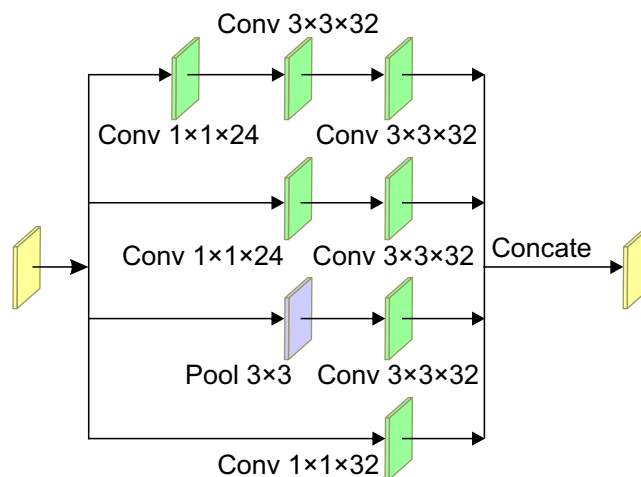


FIGURE 3.5: The architecture of the inception layer.

Each branch uses one to three convolution layers with kernel sizes of 1×1 or 3×3 . Following each convolution layer is the batch normalization and the ReLU activation function. In addition, the second branch also uses the max pooling layer to extract feature that is different from the remaining branches. With the idea of

expanding the network width using multiple scales, this module increases the receptive field for the network. The feature map has dimensions of 32×32 after being processed by this module. That means it still maintains the scaling of the dimensions in the input feature map and provides more rich information to the feature map.

3.1.3 Convolutional Triplet Attention Module

The Convolutional Triplet Attention module (CTA) (Misra et al., 2020) consists of three branches that are shown in Figure 3.6.

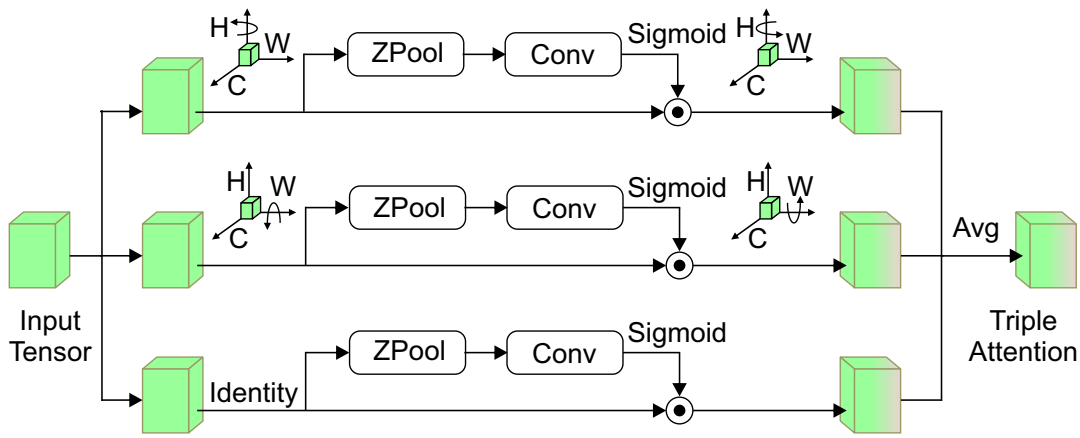


FIGURE 3.6: The architecture of the CTA module.

It assumes that the input tensor is $A \in \mathbb{R}^{C \times H \times W}$ (C: channel, H: height, W: width) and passes it through each branch in this module. The first branch considers the interaction between the width dimension and the channel dimension of the feature map by applying a 90° anti-clockwise tensor rotation along the H -axis and the rotated tensor annotated by $A_1^1 \in \mathbb{R}^{W \times H \times C}$. After that, A_1^1 go through the $ZPool$ layer (Z means the zeroth dimension of the tensor) and the dimension of A_1^1 is reduced to $A_1^2 \in \mathbb{R}^{2 \times H \times C}$. Then A_1^2 is passed to a standard convolution layer of kernel size 1×1 followed by the batch normalization layer, which generates the output tensor of $(1 \times H \times C)$. This output goes through the sigmoid function σ to obtain the attention weights and applies them to the original feature map A_1^1 using the channel element-wise multiplication operation. Then the attention feature map rotates 90° clockwise along the H -axis to get the output feature map, which is the same shape as the input feature map A .

Similarly, the second branch has exactly the same architecture as the first branch except that it focuses on the relationship between the height dimension and the channel dimension. Hence, it applies a 90° tensor rotation to the W -axis.

The last branch also uses the architecture of the first two branches but does not use tensor rotation. The results from the three branches were aggregated by simple averaging. Therefore, the final output of this module is an attention feature map of size $(C \times H \times W)$. The process of the Convolutional Triple Attention module can be presented as:

$$\mathcal{F}_{CTA} = \frac{1}{3}(\Re(A_1^1 \sigma(\phi_1(A_1^2))) + \Re(A_2^1 \sigma(\phi_2(A_2^2))) + A \sigma(\phi_3(A_3^2))), \quad (3.3)$$

where σ is the sigmoid activation function. ϕ_1, ϕ_2 , and ϕ_3 are standard convolution layers with 1×1 kernel size followed by the batch normalization layer. \Re is 90° clockwise rotation to retain original input shape.

The $ZPool$ layer is presented as follows:

$$ZPool(A) = [Max_{0d}(A), Avg_{0d}(A)], \quad (3.4)$$

in which, $0d$ is 0-th dimension which the max pooling (Max) and average pooling (Avg) layers apply in the concatenation operation $[\cdot]$. The $ZPool$ layer takes an input tensor of shape $(C \times H \times W)$ and generates an output tensor of shape $(2 \times H \times W)$.

This module is embedded right after the inception module and it allows the model to selectively focus on the most informative channels, spatial locations, and feature maps. This can lead to better feature representations and serves for more accurate predictions in the next module.

3.1.4 Detection Module

Firstly, this module uses four convolution layers, each with kernel sizes of 1×1 and 3×3 , to further reduce the dimension of the feature maps and produce four feature maps of $32 \times 32 \times 128$, $16 \times 16 \times 256$, $16 \times 16 \times 128$, and $8 \times 8 \times 256$. From these output feature maps, the network only chooses two feature maps with dimensions of $16 \times 16 \times 256$ and $8 \times 8 \times 256$ as inputs for the detection module to detect eyes at two

different levels. Detectors apply two sibling convolution layers for classification and bounding box regression. To predict a bounding box, these detectors utilize different predefined square anchors with sizes of 16, 24, and 32 for small eyes, 40 for medium eyes, and 48 for large eyes. The regression head generates a four-dimensional vector (x, y, w, h) as the offset of the bounding box location. The classification head generates a two-dimensional (eye or not-eye) vector as the label classification.

This bounding box regression process applies the parameterization of four coordinates for the predicted bounding box, anchor bounding box, and ground-truth bounding box (Ren et al., 2016). The equations of bounding box regression are shown as follows:

$$\begin{aligned}
 t_x &= (x - x_a)/w_a, t_y = (y - y_a)/h_a, \\
 t_w &= \log(w/w_a), t_h = \log(h/h_a), \\
 t_x^* &= (x^* - x_a)/w_a, t_y^* = (y^* - y_a)/h_a, \\
 t_w^* &= \log(w^*/w_a), t_h^* = \log(h^*/h_a),
 \end{aligned} \tag{3.5}$$

where (x, y) denotes the center coordinate, w denotes the width and h denotes the height of the bounding box. The parameters x , x_a , and x^* are presented for the predicted bounding box, anchor bounding box, and ground-truth bounding box respectively (similar to the parameters in y, w, h). The geometry of the bounding box regression process is shown in Figure 3.7.

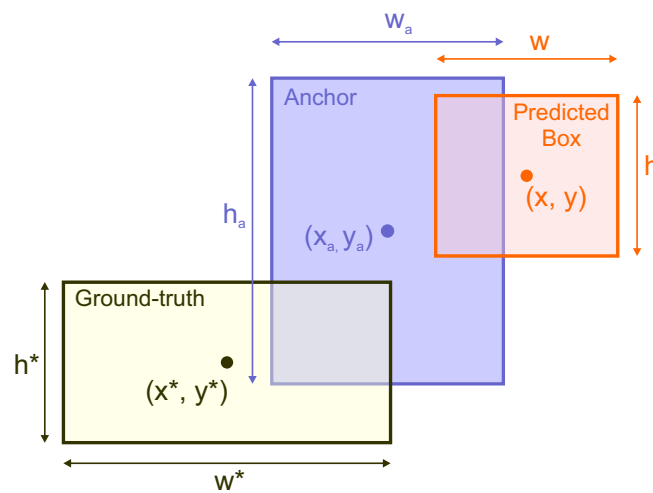


FIGURE 3.7: The geometry of the predicted bounding box, anchor bounding box, and ground-truth bounding box.

The requirement of the object detection task is selecting only the most suitable

bounding box for an object. This can be solved by applying the NMS method. The NMS is a post-processing algorithm commonly used in object detection and computer vision applications. The main purpose is to remove redundant or overlapping bounding boxes that are produced by object detection models. The algorithm works by first sorting all the detected bounding boxes based on their confidence scores. Then, it iteratively selects the bounding box with the highest confidence score and removes all other bounding boxes that have a significant overlap with it. The overlap is measured using the IoU metric, which computes the ratio of the intersection area between two bounding boxes and the union area. The process continues until all bounding boxes have been examined, and only the non-overlapping ones with the highest confidence scores are retained. The pseudo-code is explained in Algorithm 1 and the NMS visualization in the eye detection is in Figure 3.8.

Algorithm 1 Non-Maximum Suppression

```

procedure NMS( $B, c$ )
   $B_{nms} \leftarrow \emptyset$ 
  for  $b_i \in B$  do
     $Discard \leftarrow False$ 
    for  $b_j \in B$  do
      if  $same(b_i, b_j) > \lambda_{nms}$  then
        if  $score(c, b_j) > score(c, b_i)$  then
           $Discard \leftarrow True$ 
        end if
      end if
    end for
    if  $Not\ Discard$  then
       $B_{nms} \leftarrow B_{nms} \cup b_i$ 
    end if
  end for
  return  $B_{nms}$ 
end procedure

```

3.2 Loss Function

The loss function consists of classification loss and regression loss. The entire loss function is defined as follows:

$$\mathcal{L}_{ed}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* \mathcal{L}_{reg}(t_i, t_i^*), \quad (3.6)$$

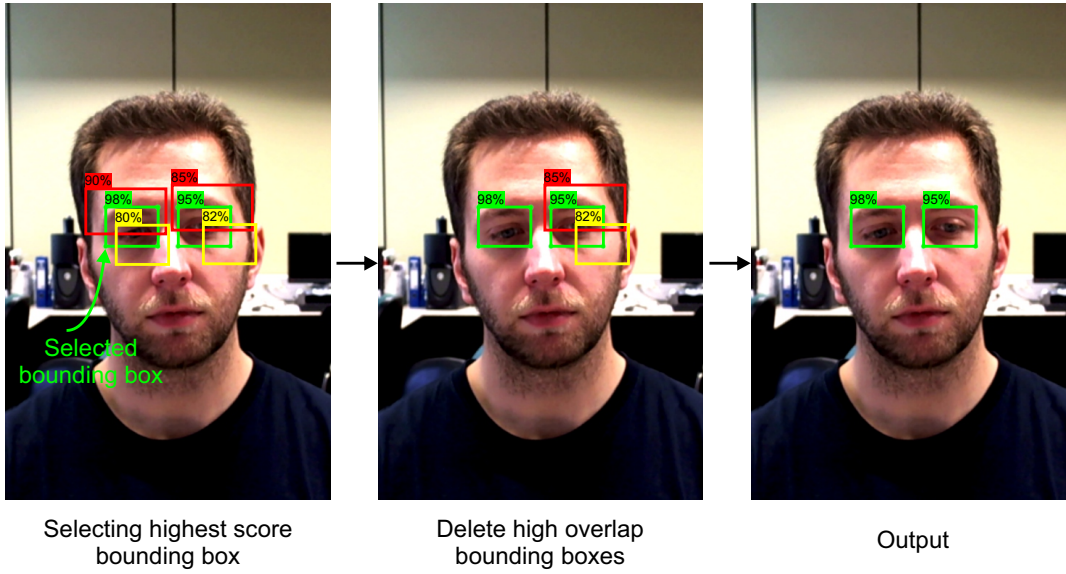


FIGURE 3.8: NMS visualization in the eye detection .

where $\mathcal{L}_{cls}(p_i, p_i^*)$ is the classification loss using the softmax-loss defined in Equation 3.7. $\mathcal{L}_{reg}(t_i, t_i^*)$ is the regression loss using the *SmoothL1* loss defined in Equation 3.8. p_i is the predicted probability and p_i^* is the ground-truth label. t_i is the center coordinates (x, y) and dimension (*height* and *width*) of the predicted bounding box. t_i^* is the ground truth bounding box of the object i . The N_{cls} is the mini-batch size, N_{reg} is the number of anchor locations and λ is the balancing parameter.

$$\mathcal{L}_{cls}(\mathcal{P}_i, \mathcal{P}_i^*) = - \sum_{i \in Pos} x_i^p \log(\mathcal{P}_i) - \sum_{i \in Neg} \log(\mathcal{P}_i^0), \quad (3.7)$$

where $x_i^p = \{0, 1\}$ is the indicator for matching between the anchor box and ground-truth box of the object i -th. \mathcal{P}_i and \mathcal{P}_i^0 are the probabilities for object and non-object classification, respectively. The \log is a natural logarithm function. The Pos is a set of positive samples and Neg is a set of negative samples.

$$SmoothL1(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (3.8)$$

3.3 Experiments

3.3.1 Proposed Datasets for Eye Detection

Along with the design of the eye detection network, this research also proposes the eye datasets containing 10,659 images with 21,318 eye labels assigned according to the PASCAL VOC dataset format (Everingham et al., 2015). The images in this dataset were collected from the BioID Face (*The BioID Face Database n.d.*), CEW (Song et al., 2014), GI4E (*GI4E - Gaze Interaction for Everybody n.d.*), FEI Face dataset (Thomaz and Giraldi, 2010), and Yale Face Dataset B (YALEB) (Georghiadis, Belhumeur, and Kriegman, 2001) dataset. The eye positions are annotated by Python code and the LabelImg annotation tool.

BioID dataset: The procedure is to select 1,521 gray-scale images with 384×286 pixel resolution from the BioID Face dataset. Each image shows the front view of the faces of 23 different people. From the center of the eye, the ground-truth bounding box is generated with a square size of 36×36 .

CEW dataset: Similarly, the CEW dataset provides 2,423 images with 100×100 pixel resolution, including 1,192 images with both eyes closed and 1,231 images with both eyes open. The ground-truth bounding box is also annotated from the center of the eye with a size of 24×24 pixels.

GI4E dataset: The GI4E contains 1,236 images from a standard camera with a resolution of 800×600 pixels in PNG format. Based on the position of the iris, the ground-truth bounding box is generated with a size of 46×46 pixels.

The three above datasets are produced by the Python code to generate ground-truth bounding boxes following the below steps and the illustration of each step is shown in Figure 3.9:

- Step 1: Determine the coordinate of the center position of the eyes from the annotation file.
- Step 2: Generate the bounding boxes from the center coordinates.
- Step 3: Create an XML file and save the top-left and bottom-right of the bounding boxes.

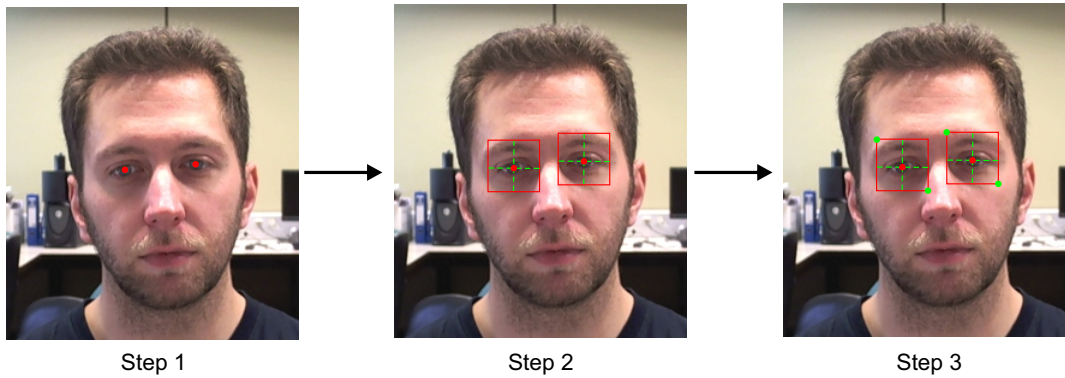


FIGURE 3.9: The annotation generation process by Python code.

FEI Face database: The FEI Face database is collected from Brazilians aged 19 to 40 with distinct appearances, hairstyles, and makeup. The images in this dataset are taken on a homogenous background with up to 180 degrees of rotation and a resolution of 640×480 pixels. This work selects 2,800 images to annotate the eye areas with the LabelImg annotation tool.



FIGURE 3.10: The interface of the LabelImg annotation tool on the FEI dataset.

YALEB dataset: The YALEB is a quite large dataset for the face detection and recognition field. It contains 16,128 images in 640×480 pixels resolution of 28 people with 9 poses and 64 different illumination conditions. A set of 2,679 images in this dataset were randomly selected and then the eye areas were also annotated using the annotation tool. In this case, the LabelImg software is also chosen to be used.

The LabelImg tool (Tzutalin, 2015) is written in the Python programming language and presents a graphical interface in Qt. The bounding boxes of the eye positions are dynamically drawn based on the actual position of the eyes in the images. The bounding box creation process is simply drawing a rectangle by dragging the mouse from the top-left corner to the bottom-right corner of the eye areas. Figure 3.10 presents the eye annotation process by the LabelImg tool.

The location coordinates (top-left and bottom-right) and other attributes of eyes in each image are generated and saved in an XML file that follows the PASCAL VOC dataset format as shown in Figure 3.11. Users can use these proposed datasets for different purposes to exploit the eye area in the computer vision field. The proposed dataset distribution is shown in Figure 3.12.

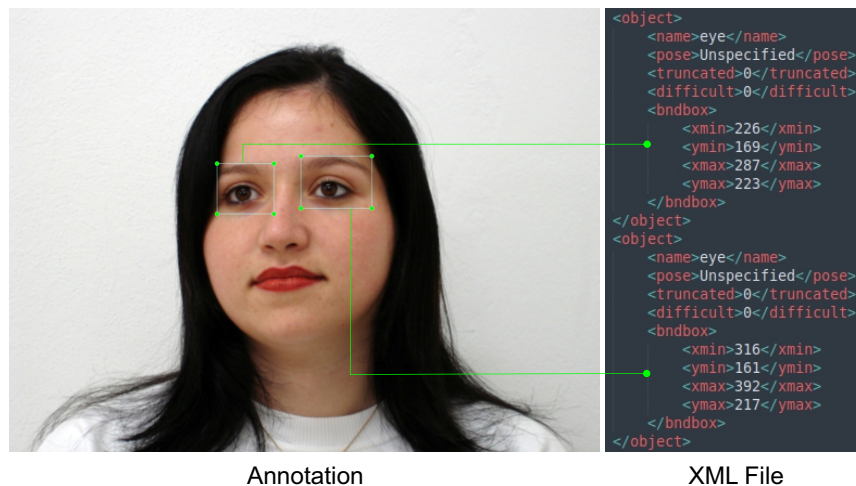


FIGURE 3.11: The sample of annotation and XML file.

3.3.2 Experimental Setup

The eye detection network is implemented by the Python programming language and the PyTorch framework. It is trained with 300 epochs and evaluated on a GeForce GTX 1080Ti 11GB GPU, 32 GB of RAM. The training process applies several configurations, such as a batch size of 16, weight decay of 5×10^{-4} , momentum of 0.9, and the learning rate is initialized by 10^{-3} and descends to 10^{-5} for training. Stochastic Gradient Descent is also used to optimize the weight during back-propagation. The balancing parameter λ is set to 10. The NMS algorithm applies a threshold of 0.5.

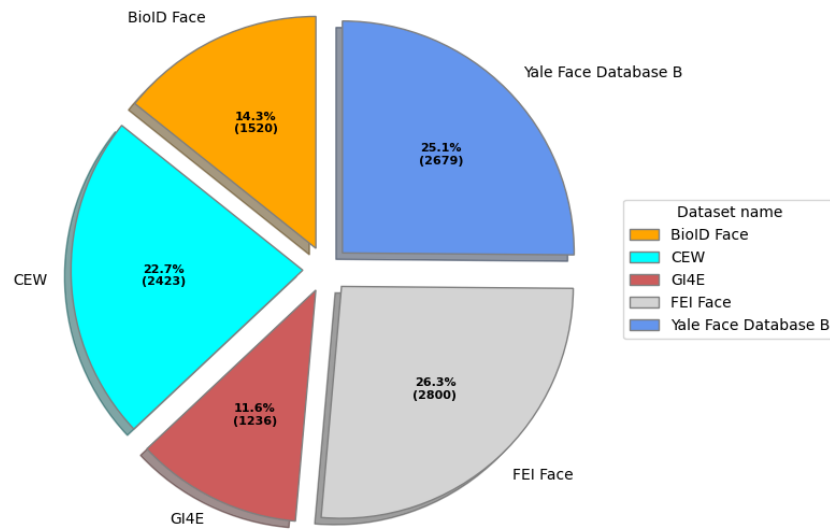


FIGURE 3.12: The proposed dataset distribution.

3.3.3 Experimental Results

This network plays an important role in determining the accuracy and efficiency of the entire eye status monitoring system. The eye detection network is trained and evaluated on proposed datasets based on BioID Face, CEW, GI4E, FEI Face, and the YALEB datasets. Additionally, to make a fair comparison with other network architectures, this work refined and retrained the FaceBoxes architecture (Zhang et al., 2017), five variants of SSD architecture (Liu et al., 2015), nine variants of YOLO architecture (Glenn Jocher, 2020), and the proposed network architectures with the change of attention modules (SE: Squeeze-and-Excitation, BAM: Bottleneck Attention Module, CBAM: Convolutional Block Attention Module). The results in Table 3.1 demonstrate that this network has superior detection capabilities compared to FaceBoxes and SSD network architectures on the CEW and FEI datasets except for all YOLO network architectures with a 96.50% and 97.14% AP, respectively. For the GI4E dataset, the proposed network achieves 98.12% AP, and it surpasses FaceBoxes and other network architectures except for SSD300, SDD512, and all YOLO network architectures. But, the number of parameters of the original SSD, YOLOV3,

TABLE 3.1: The comparison result of eye detection network on individual proposed datasets.

Model	Input image	Parameters	GFLOPs	Average Precision (%)				
				GI4E	BioID	CEW	FEI	YALEB
Mobile architectures								
FaceBoxes	128×128	844,610	0.40115	96.58	98.23	95.95	96.56	99.70
MobileNetV3-SSD	300×300	2,141,372	0.52696	75.88	90.77	90.85	90.13	99.89
MobileNetV2-SSD	300×300	3,051,928	0.65919	82.38	90.82	90.87	89.67	99.90
MobileNetV1-SSD	300×300	3,990,680	1.24000	83.83	80.51	90.86	89.48	99.91
SSD300	300×300	23,745,908	62.82000	99.90	90.90	90.30	90.90	90.90
SSD512	512×512	23,745,908	180.61000	100	90.80	90.70	81.70	90.00
YOLOV3-Tiny	416×416	8,669,876	12.90000	99.98	98.97	97.62	98.46	99.54
YOLOV3-SPP	416×416	62,573,334	155.70000	100	99.00	97.97	98.32	99.93
YOLOV3	416×416	8,669,876	154.90000	100	97.90	99.73	99.40	99.96
YOLOV4-Tiny	320×320	3,061,636	6.40000	99.90	99.70	99.40	99.70	99.70
YOLOV4	320×320	60,426,006	131.10000	99.70	99.60	99.20	99.96	99.96
YOLOV5s	320×320	7,063,542	16.40000	99.70	99.60	99.60	99.60	99.60
YOLOV5m	320×320	21,056,406	50.40000	99.70	99.60	99.30	99.60	99.70
YOLOV5l	320×320	46,631,350	114.20000	99.70	99.60	99.30	99.60	99.60
YOLOV5x	320×320	69,117,998	170.50000	99.70	99.60	99.30	99.60	99.60
Our architectures								
Proposed	128×128	965,186	0.4934	98.12	98.35	96.50	97.14	99.66
Our (SE)	128×128	967,070	0.4926	97.30	98.21	95.12	97.02	99.60
Our (BAM)	128×128	969,343	0.4948	98.05	99.05	95.04	97.01	99.80
Our (CBAM)	128×128	969,354	0.4926	90.84	97.31	94.78	97.10	99.68

YOLOV4, and YOLOV5m are 24.6, 8.98, 62.61, and 21.82 times larger than the proposed network architecture, respectively. For the BioID dataset, this experiment achieved an approximate AP when replacing the Convolutional Triplet Attention module by BAM with an accuracy of 98.35% and 99.05% AP, respectively. Nevertheless, the BAM has more than 4,157 network parameters. It also outperformed the other compared network architectures except for YOLO network architectures. In the end, the proposed network reached a 99.66% AP on the YALEB dataset. This result is comparable to the FaceBoxes network, outperforming both SSD300 and SSD512, and it is better than several YOLO network architectures. However, it is slightly weaker than the other proposed network architectures. The reason is the YALEB dataset contains a large number of black and white images with many different lighting conditions levels. This makes it difficult for lightweight or shallow architectures to distinguish small objects in the human eye regions. In terms of computational complexity, the proposed network is only marginally larger than the FaceBoxes network at 0.09 GFLOPs, greatly smaller than the SSD and YOLO network families (from 13 times (YOLOV4-Tiny) to 336 times (SSD512)). The qualitative results of the eye detection network are shown in Figure 3.13.

Though, sometimes the proposed network confuses and detects wrong in other



FIGURE 3.13: The qualitative results on five proposed datasets.



FIGURE 3.14: The several mistakes in detection results on five proposed datasets.

areas such as the objects in the background image or several parts of the face that are similar in structure to the eye like the mouth, and forehead. It also may not detect the eye area due to the light, color, contrast, and overlap conditions. Several mistake-detection samples are shown in Figure 3.14.

3.4 Ablation Studies

To evaluate the effectiveness of different components in the eye detection network, each module was omitted during training and evaluation, then the results with the entire eye detection network were compared. The results in Table 3.2 demonstrate that the use of the Convolution Triplet Attention module is necessary to increase the

prediction ability (adding 1.71% to the AP) and maintain the parameters of the entire network. The omission of the inception module reduces a lot of network parameters (0.227 million parameters) but the precision is not significantly reduced (by only 0.2% of the AP). Therefore, the inception module may be omitted when designing the network to optimize the number of parameters. The CReLU module increases the efficiency of the feature extractor so when omitted it can reduce the AP by nearly 1%. Finally, using more than two detectors does not increase predictability, on the other hand, it increases the number of network parameters.

TABLE 3.2: Ablation studies of eye detection network on CEW dataset. The red color indicates the best result.

Modules	Proposed network				
Convolutional Triple Attention		✓	✓	✓	✓
Inception	✓		✓	✓	✓
CReLU	✓	✓		✓	✓
Detectors	2	2	2	3	2
Parameters (Million)	0.965	0.738	0.949	1.072	0.965
AP (%)	94.79	96.30	95.54	95.95	96.50

Chapter 4

Eye Classification Network

Eye classification is the final stage in the driver eye status monitoring system. The network architecture is designed by taking advantage of the basic layers used in CNN networks and improving several layers to optimize network parameters and eye status discrimination ability. The eye classification network is described in detail as shown in Figure 4.1.

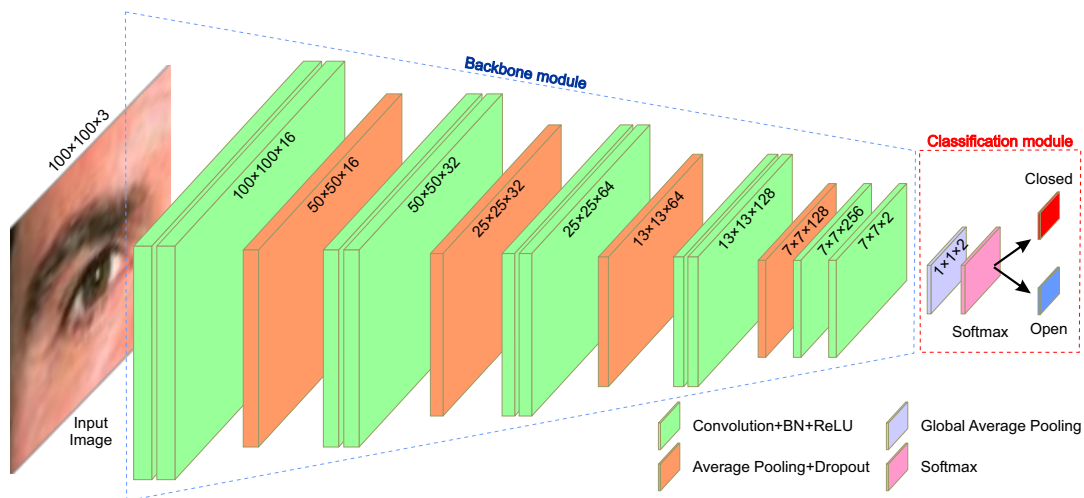


FIGURE 4.1: The proposed eye classification network architecture.

4.1 Network Architecture

This network is simply designed based on the basic usage of convolution, average pooling layers, and the softmax function to classify eye status. The eye classification network is divided into two main modules: backbone and classification.

4.1.1 Backbone Module

The backbone module uses four convolution blocks and two separate convolution layers. Each of these blocks contains two standard convolution layers and one average pooling layer followed by batch normalization (BN) and the ReLU activation function.

Average pooling layer: The working principle of the average pooling layer is exactly the same as the max pooling layer (which was explained in Chapter 3), but the only difference is that it outputs by averaging the values in each window. The operation of the average pooling layer is depicted as an example in Figure 4.2.

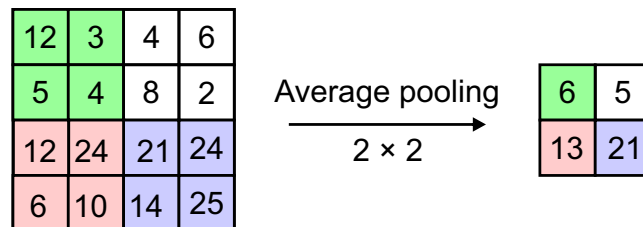


FIGURE 4.2: The operation of the average pooling layer.

Batch normalization (BN): The BN is a technique used in deep learning neural networks to improve the training process and performance of the model. The BN works by normalizing the input values of a layer by subtracting the mean and dividing by the standard deviation of the batch. This process is applied for each batch of data during training, which helps to stabilize the distribution of inputs and the model can easier to learn the correct weights. Assume \mathcal{M} is a minibatch and let $x \in \mathcal{M}$ be an input to batch normalization. The batch normalization is defined as follows:

$$BN(x) = \gamma \cdot \frac{x - \mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}}, \quad (4.1)$$

where $\mu_{\mathcal{M}}$ is the mean and $\sigma_{\mathcal{M}}$ is standard deviation of the minibatch \mathcal{M} .

ReLU activation function: From Equation 3.1 in Chapter 3 can see that the output of the ReLU activation function is the maximum of 0 value and the input value. This means that the output is 0 value for any negative input and the output is equal to the input value for any positive input. This activation function is computationally efficient, allows for faster training, and avoids the over-fitting problem of the network.

The kernel sizes of the convolution layers used in this module are 7×7 , 5×5 , and 3×3 respectively according to the depth of the network. The backbone module reduces the dimension of the input image from 100×100 to 7×7 .

4.1.2 Classification Module

The classification module consists of two components: the global average pooling (GAP) layer and the softmax function. In global average pooling, the spatial dimensions of the feature maps are quickly reduced to a single value by taking the average of all the values in each channel. This results in a one-dimensional vector with the same number of channels as the input feature maps. Following that, the output feature map from the backbone module is further reduced by the global average pooling layer to 1×1 , with the two channels corresponding to the two classes in the dataset. Figure 4.3 illustrates the operation of the global average pooling layer.

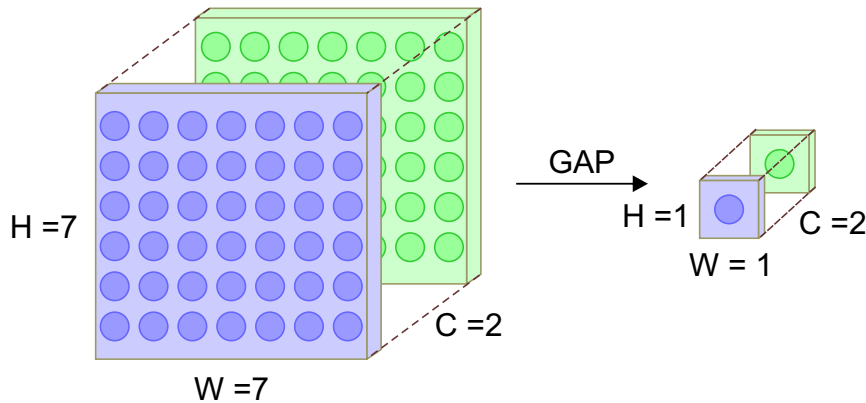


FIGURE 4.3: The global average pooling layer.

Then, it applies the softmax function to calculate the probability of occurrence of the classes. The equation for the softmax function is:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}}, \quad (4.2)$$

where x_i is the i -th element of the input vector, and k is the total number of elements in the vector. The e^{x_i} and e^{x_j} are standard exponential functions for the input and output vector, respectively. The softmax function takes an input vector x and returns a vector of the same length, where each element in the output vector is a probability value between 0 and 1 that sums up to 1.

The replacing of the fully connected layers in traditional classifiers with only one GAP layer to optimize network parameters and further avoid overfitting issues. This is an important improvement in the proposed eye classification network.

4.2 Loss Function

The classifier uses the Category cross-entropy loss function (Fawaz et al., 2018) to compute the loss during training. This loss is shown as follows:

$$\mathcal{L}_{ec} = - \sum_{i=0}^1 t_i \cdot \log(\mathcal{P}_i), \quad (4.3)$$

where i is the index of each class ($i = 0$ to 1), t is the target indicator ($t = 0$ or $t = 1$), \log is natural logarithm function, and \mathcal{P} is the predicted probability of class i -th.

4.3 Experiments

4.3.1 Datasets

For the eye classification network, the Closed Eyes In The Wild (CEW) (Song et al., 2014) and Media Research Lab (MRL) Eye (Fusek, 2018) datasets are used for training and evaluation. The CEW dataset consists of 4,846 images of 24×24 pixels resolution with 2,384 closed eye labels and 2,462 open eye labels. To enrich the dataset, dataset augmentation methods are applied during training, such as vertical flip, contrast, and brightness changes. MRL Eye dataset contains 84,898 images of 86×86 pixels resolution collected from 37 different people (33 males and 4 females) by three sensors (Intel RealSense RS 300, IDS Imaging sensor, and Aptina sensor). Two datasets are separated into 80% for training and 20% for evaluation phases.

4.3.2 Experimental Setup

The eye classification network is built using the Python programming language and the Keras framework. It is trained and evaluated with basic settings for the classification task on a GeForce GTX 1080Ti 11GB GPU, with 32 GB of RAM. Specifically,

the network goes through 200 epochs with the Adam optimization method and a batch size of 16. The learning rate is initialized to 10^{-4} and decreases after 10 epochs with a factor of 0.75 if the accuracy does not improve.

4.3.3 Experimental Results

The eye classification network achieves accuracies of 97.53% and 98.52% on the CEW and MRL Eye datasets, respectively. It outperforms all networks with just 632,978 network parameters and 0.0067 GFLOPs. The qualitative results are shown in Figure 4.4.

TABLE 4.1: The comparison result of the eye classification network with popular classification networks on CEW and MRL Eye Datasets. The red color indicates the best competitors.

Network	Parameters	GFLOPs	Accuracy (%)	
			CEW	MRL Eye
SqueezeNet	256,818	0.0008	50.72	98.32
Proposed	632,978	0.0067	97.53	98.52
MobileNetV2	3,571,778	0.0026	67.11	68.53
MobileNet	4,280,514	0.0021	82.27	81.80
VGG13	7,052,738	0.0011	96.29	98.33
DenseNet121	8,089,154	0.0021	79.59	87.50
VGG16	15,242,050	0.0011	94.33	97.47
LeNet	15,653,072	0.1869	96.70	98.08
VGG19	20,551,746	0.0011	94.12	97.18
Xception	22,961,706	0.0011	62.99	97.53
ResNet50	23,591,810	0.0042	94.85	97.50
InceptionV3	23,903,010	0.0042	76.60	79.38
AlexNet	67,375,938	1.3500	96.71	98.42

Specifically, when compared with the AlexNet architecture (the best accuracy competitor), the network parameters and computational complexity of AlexNet are 106 and 201 times larger but it only achieves an accuracy of 96.71% and 98.42%, which is lower than the proposed network 0.82% and 0.10% on the CEW and MRL Eye datasets, respectively.

For the SqueezeNet architecture (the best network scale competitor), the network parameters and computational complexity of the proposed network are nearly 2.5 and 0.84 times larger, but the accuracy of the proposed network is nearly twice as large on the CEW dataset and 0.20% better on the MRL Eye dataset. Table 4.1 shows a detailed comparison of results between the proposed eye classification approach

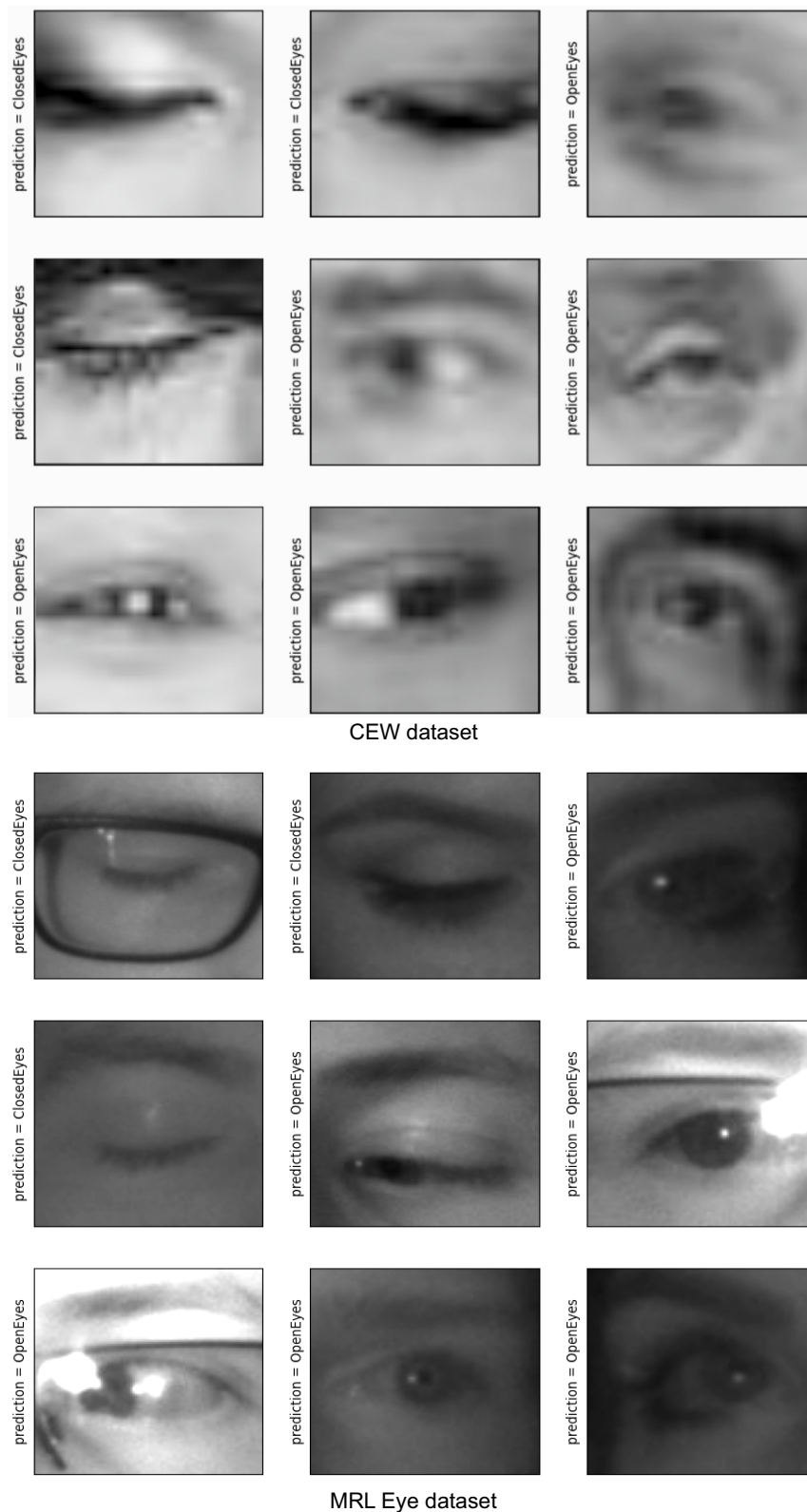


FIGURE 4.4: The qualitative results of proposed eye classification network on CEW and MRL Eye datasets.

and popular classification networks. The confusion matrices in Figure 4.5 prove the balance in closed and open eye classification on the CEW and MRL Eye datasets.

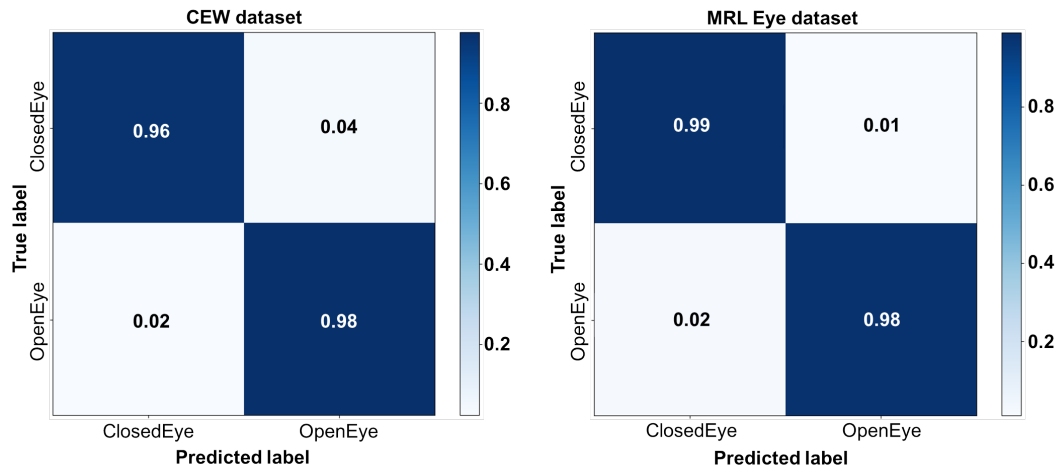


FIGURE 4.5: The confusion matrices on CEW and MRL Eye datasets.

4.4 Ablation Studies

This work conducts ablation studies on the optimization method and the classification module to evaluate the proposed eye classification network’s outstanding advantages.

For ablation study 1, when replacing the Adam optimizer with the AdamW optimizer, the accuracies achieved on the CEW and the MRL Eye datasets are 97.50% and 98.44%, respectively. The experimental results show that Adam is better than AdamW (0.03% \uparrow of accuracy on the CEW dataset and 0.08% \uparrow of accuracy on the MRL Eye dataset) in optimizing the proposed eye classification network. The performance comparison of Adam and AdamW optimizers is shown in Table 4.2.

TABLE 4.2: The comparison result of the eye classification network with different optimizers on CEW and MRL Eye Datasets.

Optimizers	Parameters	GFLOPs	Accuracy (%)	
			CEW	MRL Eye
AdamW	632,978	0.0067	97.50	98.44
Adam	632,978	0.0067	97.53	98.52

For ablation study 2, this work replaces the GAP with fully connected (FC) layers using only one hidden layer (1024 network nodes) for training and evaluation in the same setting. With the use of FC in the classification module, the eye classification network has accuracies of 96.29% and 98.34%, respectively. This result is worse than the eye classification network that uses GAP (1.24% \downarrow of accuracy on the CEW dataset and 0.18% \downarrow of accuracy on the MRL Eye dataset). In addition,

FC also increases a lot of network parameters (103426 parameters \uparrow) and computational complexity (0.6588 GFLOPs \uparrow). Therefore, the use of GAP in the proposed eye classifier network contributes greatly to the optimization of network parameters and computational complexity, aiming to apply on low computational devices. The comparison result is shown in Table 4.3.

TABLE 4.3: The comparison result of eye classification networks with the GAP and fully connected (FC) layers on CEW and MRL Eye datasets.

Methods	Parameters	GFLOPs	Accuracy (%)	
			CEW	MRL Eye
FC	736,404	0.6655	96.29	98.34
GAP	632,978	0.0067	97.53	98.52

Chapter 5

Real-time Eye Status Monitoring

After training and evaluating each individual network on the respective datasets, the networks are integrated with a nano version of the YOLO5Face detector (YOLO5nFace) (Qi et al., 2021) into a complete eye status monitoring system. This system is tested in real-time and driving simulation videos, as described by Figure 5.1 and the block diagrams in Figure 5.4.

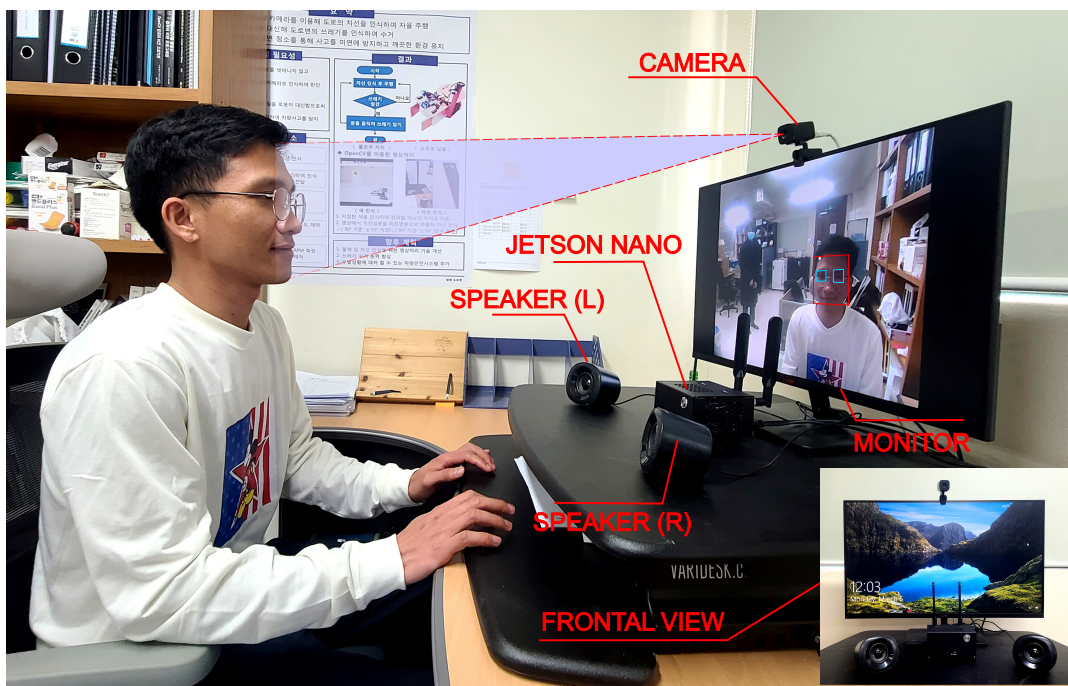


FIGURE 5.1: The real-time testing setting with all devices.

5.1 YOLO5Face Detector

This face detector is developed based on the architecture of the YOLOv5 network with many improvements for face detection, named YOLO5Face. The network is

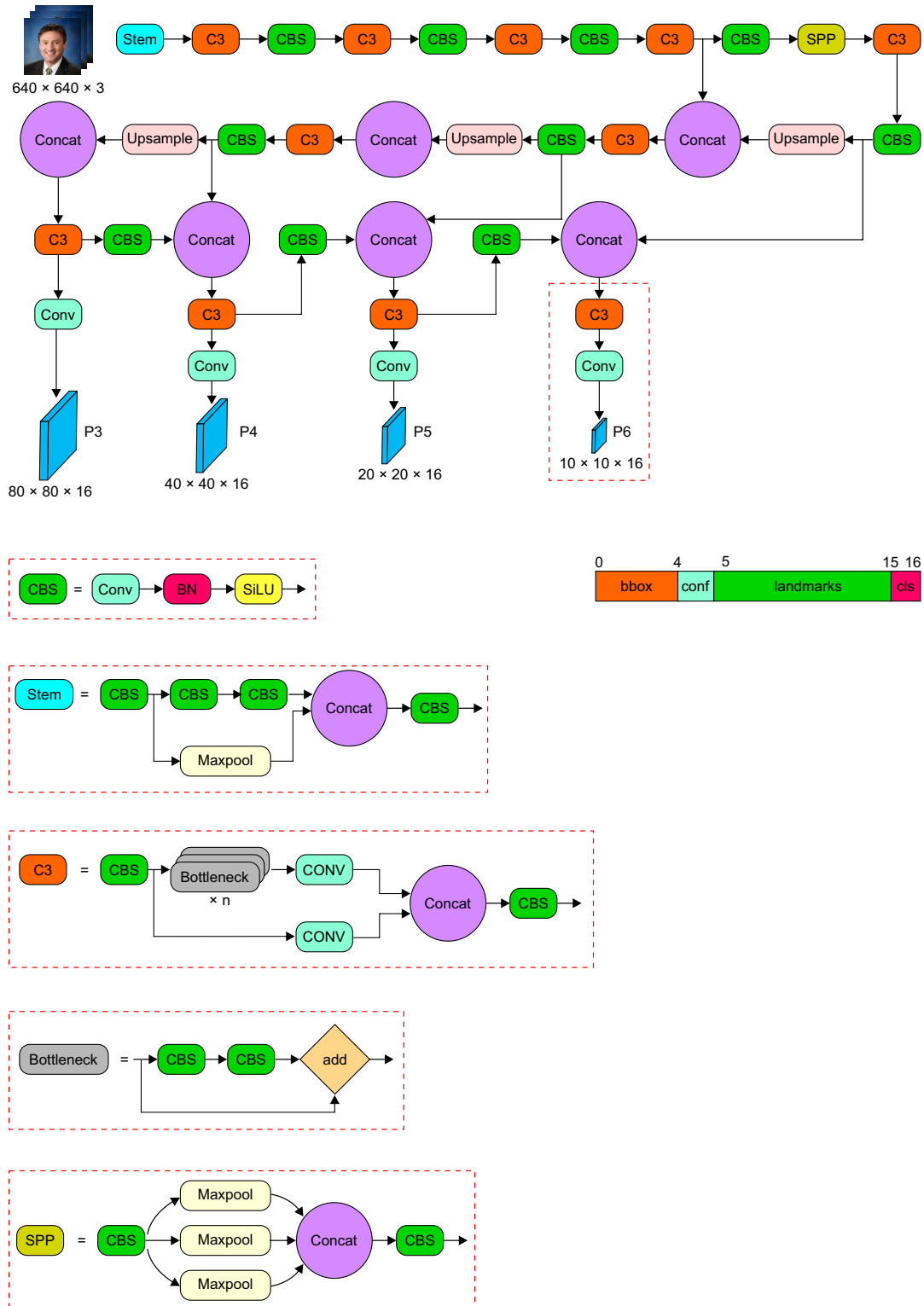


FIGURE 5.2: The YOLO5Face architecture and sub modules.

trained and evaluated on the WIDER FACE dataset and it achieves state-of-the-art performance in almost all the easy, medium, and hard subsets. Especially, the authors also design several tiny networks with lightweight backbones to implement in embedding and mobile devices.

5.1.1 Network Architecture

Following the original architecture of YOLOv5, this proposed network is divided into three main parts: backbone, neck, and detection head modules as shown in Figure 5.2. However, the authors have conducted many improvements to suit the face detection task and increase the performance on low-computing devices. Such modifications can be listed as follows:

- The landmark regression is added to the YOLOv5 network and used the Wing loss (Feng et al., 2017) to calculate during the training. This is an extra supervision that helps increase face detection accuracy.
- The Focus layer in YOLOv5 is replaced with a Stem block structure (Wang et al., 2018). It increases the generalization capability and reduces the computational complexity while still maintaining performance.
- The kernel size of max pooling layers in the SPP block (He et al., 2014) is revised from 13×13 , 9×9 , and 5×5 to 7×7 , 5×5 , and 3×3 , respectively. These smaller kernels help to detect small faces easier and significantly reduce the number of network parameters.
- This work leverages three detection heads from YOLOv5 and adds one more head at the P6 output feature map with a stride of 64. It increases the ability to detect large faces. This is easily ignored by other studies that only focus only on detecting small faces.
- The implement removes several data augmentation methods that are not appropriate for face detection, including up-down flipping and Mosaic. It recognizes random cropping method is useful for performance improvement.
- This work designs nine model versions with different backbone networks as shown in Table 5.1.

5.1.2 Dataset

All of the modified networks are trained and evaluated on the WIDER FACE dataset (Yang et al., 2015). This is the largest dataset for face detection. This dataset consists of 32,203 images and 393,703 annotated faces. The images in the WIDER FACE

TABLE 5.1: The variants of YOLO5Face network (Qi et al., 2021).

Model	Backbone	(D, W)	With P6
YOLO5sFace	YOLOv5-CSPNet	(0.33, 0.50)	NO
YOLO5s6Face	YOLOv5-CSPNet	(0.33, 0.50)	YES
YOLO5mFace	YOLOv5-CSPNet	(0.50, 0.75)	NO
YOLO5m6Face	YOLOv5-CSPNet	(0.50, 0.75)	YES
YOLO5lFace	YOLOv5-CSPNet	(1.0, 1.0)	NO
YOLO5l6Face	YOLOv5-CSPNet	(1.0, 1.0)	YES
YOLO5x6Face	YOLOv5-CSPNet	(1.33, 1.25)	YES
YOLO5nFace	ShuffleNetv2	-	NO
YOLO5nFace-0.5	ShuffleNetv2-0.5	-	NO

dataset are very close to real-life scenes and contain many challenges with various scales, poses, expressions, lighting, and occlusions. The dataset is divided into 50%, 10%, and 40% for training, validation, and testing phases.

5.1.3 Experimental Results

The detection ability of the YOLO5Face network is evaluated through two groups of comparison models: large and small models. In the first group, the YOLOv5x6Face network achieves 96.67%, 95.08%, and 86.55% of mAP on easy, medium, and hard subsets, respectively. These results outperform the existing comparative networks on all three subsets. In the second group, the YOLOv5nFace network achieved achieves 93.61%, 91.54%, and 80.53% on the three subsets, respectively. This result is a little bit worse performance than the Sample and Computation Redistribution for Efficient Face Detection (SCRFD) network on easy (0.17% ↓) and medium (0.62% ↓) subsets. But on the hard subset, YOLOv5nFace is 2.66% of mAP better than the SCRFD network. Compare to others in this group, YOLOv5nFace still performs better with only 1.73 million network parameters and 2.11 GFLPS. That's why this study uses YOLOv5nFace for the real-time eye status monitoring system. The comparison results are presented in Table 5.2.

5.2 Integrated System

The overall proposed driver eye status monitoring system consists of a face detection network (the trained YOLO5nFace network on the WIDER FACE dataset), a proposed eye detection network (the trained eye detection network on the CEW

TABLE 5.2: Comparison of YOLO5Face networks and existing face detectors on the WIDER FACE validation dataset (Qi et al., 2021).

Detector	Backbone	Easy	Medium	Hard	Params (M)	GFLOPs
DSFD	ResNet152	94.29	91.47	91.47	91.47	91.47
RetinaFace	ResNet50	94.92	91.90	64.17	29.50	37.59
HAMBox	ResNet50	95.27	93.76	76.75	30.24	43.28
TinaFace	ResNet50	95.61	94.25	81.43	37.98	172.95
SCRFD-34GF	Bottleneck ResNet	96.06	94.92	85.29	9.80	34.13
SCRFD-10GF	Basic ResNet	95.16	93.87	83.05	3.86	9.98
YOLOv5sFace	YOLOv5-CSPNet	94.33	92.61	83.15	7.08	5.78
YOLOv5s6Face	YOLOv5-CSPNet	95.48	93.66	82.80	12.39	6.28
YOLOv5mFace	YOLOv5-CSPNet	95.30	93.76	85.28	21.06	18.15
YOLOv5m6Face	YOLOv5-CSPNet	95.66	94.10	85.20	35.49	19.77
YOLOv5lFace	YOLOv5-CSPNet	95.90	94.40	84.50	46.632	41.61
YOLOv5l6Face	YOLOv5-CSPNet	96.38	94.90	85.88	76.67	45.28
YOLOv5x6Face	YOLOv5-CSPNet	96.67	95.08	86.55	141.16	88.67
SCRFD-2.5GF	Basic ResNet	93.78	92.16	77.87	0.67	2.53
SCRFD-0.5GF	Depth-wise Conv	90.57	88.12	68.51	0.57	0.51
RetinaFace	MobileNet0.25	87.78	81.16	47.32	0.44	0.80
FaceBoxes	-	76.17	57.17	24.18	1.01	0.28
YOLOv5n	ShuffleNetv2	93.61	91.54	80.53	1.73	2.11
YOLOv5n0.5	ShuffleNetv2-0.5	90.76	88.12	73.82	0.45	0.57

dataset), a proposed eye classification network (the trained eye classification network on the CEW dataset), a camera (TGCAM-2000STAR), a set of the mini speaker (INKEL speaker), and an *Intel*[®] *Core*[™] i7-4770 CPU @ 3.40GHz with 8 GB of RAM or a 128-core Nvidia Maxwell GPU with 4 GB of RAM (Jetson Nano device). Figure 5.3 shows the hardware used in the real-time testing setting.

The operation of the real-time eye status monitoring system is described by the block diagram in Figure 5.4. Firstly, the face detection network will detect the face position in the videos, crop, and resize it to 128×128 pixels as input to the eye detection network. Next, the eye detection network will detect the eye position in the previously cropped face areas, crop, and resize to 24×24 pixels. Finally, based on these cropped eye regions, the eye classification network will perform the classification of eyes with two statuses that are closed or open. If the closed eye exceeds the time threshold (in this case sets the time threshold to 2 seconds), the speaker will sound a warning to the driver. The Non-Maximum Suppression (NMS) algorithm is also used in the first two stages to minimize the generation of redundant bounding boxes and ensure the generation of one bounding box for one face and two bounding boxes for two eyes on that face.



FIGURE 5.3: The hardware is used in the real-time eye status monitoring system.

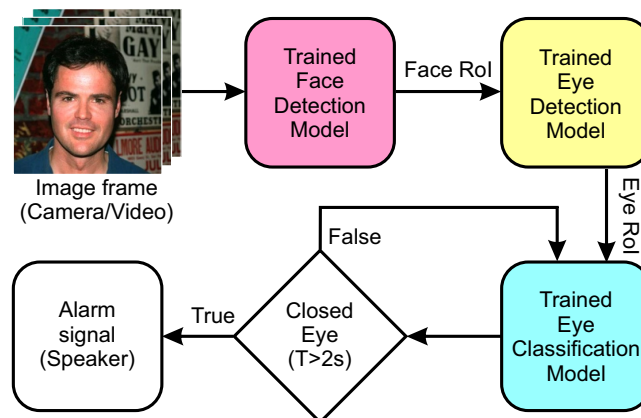


FIGURE 5.4: Block diagram of the proposed driver eye status monitoring system.

5.3 Real-time Analysis

The testing system uses the VGA and HD live-stream videos obtained from the camera connected to a PC or Jetson Nano device, and the FHD videos captured from a cellphone and downloaded from YouTube. The distance from the camera to the participant's face is less than 0.5 meters. The video resolutions used in this test are

Video Graphics Array (VGA: 640×480 pixels), High Definition (HD: 1280×720 pixels), and Full High Definition (FHD: 1920×1080 pixels). The report in the experiment with three attendances, including two males and one female. As a result, the real-time system correctly predicts the eye status with different head poses and situations, like wearing glasses, facemask, hat, wearing both hat and facemask (Figure 5.5 and Figure 5.6), the changes of illumination, infrared video, and wearing different kind of glasses (Figure 5.7).



FIGURE 5.5: The qualitative results of the real-time eye status monitoring system with VGA video live-stream on Jetson Nano device with three participants.

The face detection network reached up to 228.82 FPS on a CPU with VGA resolution and this is reduced to a minimum of 76.73 FPS on a Jetson Nano device with FHD resolution. Likewise, the eye detection network reached 39.59 FPS on a CPU with VGA resolution and decreased to a minimum of 30.13 FPS on a Jetson Nano device with HD resolution. In contrast, the eye classification network maintained speeds from 68.12 FPS on a CPU with HD resolution to 77.31 FPS on a Jetson Nano device with FHD resolution. The classification ability of the eye classification network on the Jetson Nano device is slightly better than that of the CPU. In total, the system reached a minimum speed of 20.23 FPS on a Jetson Nano device with FHD resolution and a maximum speed of 33.12 FPS on a CPU with VGA resolution. The detailed



FIGURE 5.6: The qualitative results of the real-time eye status monitoring system with VGA video live-stream on CPU-based PC with the single participant.

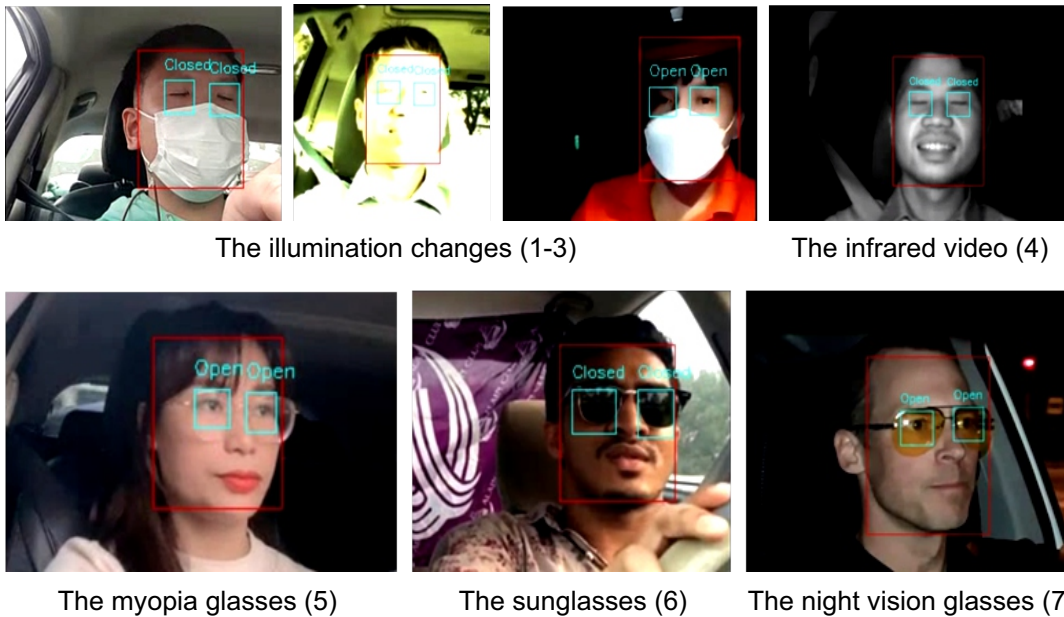


FIGURE 5.7: The qualitative results of the real-time eye status monitoring system with VGA video recording in the car on CPU-based PC.

execution speeds on the CPU and Jetson Nano devices are shown in Table 5.3.

This research also compares the speed of the proposed system and other methods in real-time testing. The experiment in (Saurav et al., 2022) is conducted with two types of devices: Nvidia Xavier (512-core Volta GPU and 64 GB of RAM) and

TABLE 5.3: The speed performance of the system in real-time testing on PC and Jetson Nano device. The red color indicates the best result. (Unit of measurement: FPS).

Device	Resolution	Face detection	Eye detection	Eye classification	Entire system
CPU	VGA	228.82	39.59	69.34	33.12
	HD	135.53	38.98	68.12	26.97
	FHD video	98.06	37.67	70.91	25.75
Jetson Nano	VGA	150.23	31.15	76.14	25.11
	HD	110.56	30.13	75.89	23.15
	FHD video	76.73	30.56	77.31	20.23

Raspberry Pi 3 (quad-core ARM Cortex-A53 CPU and 1 GB of RAM) with Intel Neural Compute Stick 2 (RPi3 + NCS2). This work achieves 62 FPS and 11 FPS, respectively. The hardware specification in Table 5.4 and the results in Figure 5.8 prove that the speed of the proposed system is fall between the two above experiments and it works well in real-time with negligible delay to accurately predict the eye status in VGA resolution.

TABLE 5.4: The hardware specification of system in (Saurav et al., 2022) and proposed system.

System	Device	Hardware specification
Saurav et al., 2022	Nvidia Xavier	512-core Volta GPU, 64 GB of RAM
Saurav et al., 2022	Pi3+NCS2	Quad-core ARM Cortex-A53 CPU, 1 GB of RAM
Our	Personal Computer	Intel® Core™ i7-4770 CPU @ 3.40GHz, 8 GB of RAM
Our	Jetson Nano	128-core Nvidia Maxwell GPU, 4 GB of RAM

The proposed driver eye status monitoring system can be highly influenced by lighting conditions, which can significantly reduce the system's ability to detect eye areas. On the other hand, the distance and angle from the camera to the driver face is a factor that increases or decreases the size of the detected face and changes the accuracy. Besides, when the driver uses sunglasses, it is not possible to identify the eye areas because most of the eye area has been obscured. In this case, the system can not recognize eye locations or predict the closed status. However, when using night vision glasses for driving, the detection results are still very accurate even when driving at night. Technically, because the system uses three stages, the execution speed is affected and depends heavily on the face detection network. In addition, the ability of the eye detection network also determines most of this speed. Therefore, shortening the system down to only one or two stages for direct eye detection and classification is necessary. This method will greatly enhance performance on real-time testing phase.

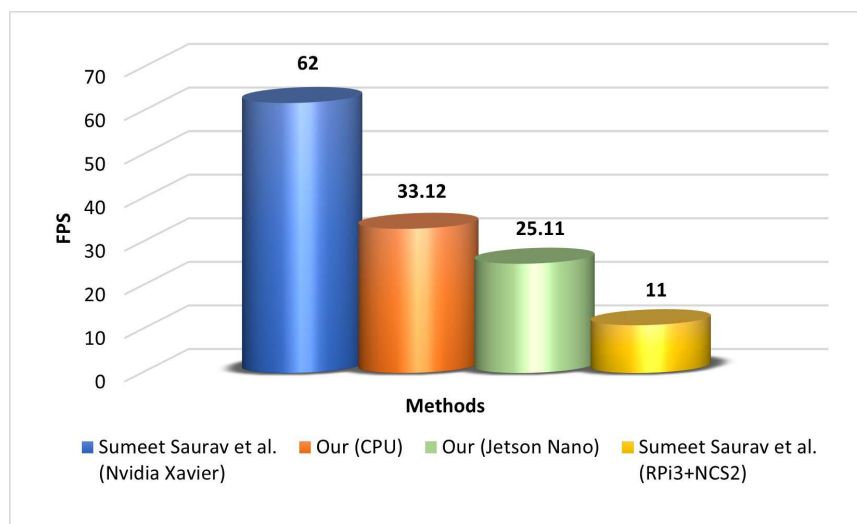


FIGURE 5.8: The speed comparison between the proposed method and others in real-time testing on VGA (640 × 480) resolution.

Chapter 6

Conclusion

6.1 Conclusions

This research aims to develop and test a real-time driver eye monitoring system based on the proposed lightweight CNN network architectures. The proposed system provides robust monitoring of the human eye in laboratory environments and on videos simulating complex driving situations. The overall system is implemented, evaluated, and explained in detail through each proposed module.

Firstly, this work designs a lightweight eye detection network with a combination of shrinking, inception, Convolutional Triple Attention mechanism, and detection modules. The shrinking module is located at the beginning of the network to extract basic eye features such as color, edge, and shape. Next, the inception module performs intermediate feature extraction using convolution and pooling operations with different kernel sizes. In this step, feature maps are extracted with various receptive fields for merging and enriching the useful information. A Convolutional Triple Attention mechanism is embedded right after the inception module to help the network focus on salient features to capture the necessary features for later detection. Ending the eye detection network are two detectors with appropriate anchor designs for predicting eye position and eye objectness scores. Realizing the limitation of datasets for human eye detection, this work also proposes new datasets with annotations conforming to the PASCAL VOC dataset format. These datasets can be used in other related fields of research. The proposed eye detection network achieves 98.12%, 98.35%, 96.50%, 97.14%, and 99.66% of AP on the GI4E, BioID, CEW, FEI, and YALEB respectively. These results are comparable with

existing detection networks and with only 965,186 network parameters and computational complexity of 0.4934 GFLOPS.

Secondly, this study continues to propose a simple and compact CNN network for the eye classification task. The architecture of this network is built on the basic components of a common classification network consisting of two main modules, the backbone network and the classification. The backbone module is a combination of alternating convolution and pooling layers followed by BN and activation functions. This module plays an important role in feature extraction and also provides high-level feature maps for the next stage. Unlike conventional classification CNNs, the classification module replaces all fully connected layers with a single GAP layer. This technique rapidly reduces the dimensionality of the feature map while preserving significant information in the feature map. It is followed by a softmax activation function to calculate the probability of the classes will be classified (closed eye or open eye). The proposed classification network is trained and evaluated on CEW and MRL Eye datasets with an accuracy of 96.71% and 98.42%, respectively. When compared with other classification CNNs, the proposed network is the outperforming method with only 632,978 network parameters and a computational complexity of 0.0067 GFLOPS.

Thirdly, a real-time eye monitoring system is integrated and tested with a novel face detection network, named YOLO5nFace. This face detection network is improved from the famous object detection network YOLOv5. The components inside YOLOv5 are optimized and redesigned to accommodate face detection tasks in various face scales and poses. On the other hand, it also meets the implementation in reality applications. The proposed system is evaluated with live stream videos and simulated videos of real driving situations. As a result, this system reaches the highest speed at 3.12 FPS and 25.11 FPS on VGA resolution with PC and Jetson Nano devices, respectively. This result proves that the proposed system can be applied to real-time systems for eye-monitoring purposes with high accuracy and negligible latency.

Finally, each experiment is carefully and objectively evaluated and analyzed for comparison with existing methods. Besides, the integrated system is also deployed on the available cheap hardware devices and considered based on the pros and cons. From there, this work suggests solutions to improve the performance of the system

in the future.

6.2 Future Works

During the research and experiment, this study recognized the limitation of eye detection and eye status classification, because it deploys an eye status monitoring system with multi-stages. In which, each stage corresponds to a separate CNN network and dataset. This method can achieve a quite large accuracy, but the speed is limited by the multi-task mechanism. This also is a major obstacle when deploying in real-time systems that require high mobility, accuracy, and speed.

To overcome the above problem, future work aims to improve the proposed system into a two-stage or single-stage system.

The two-stage system direction will focus on better designing the proposed eye detection network and combining it with new techniques such as Vision Transformer, Dilated convolution, and Path Aggregation Network (PAN) architecture, etc to directly detect eye regions (small regions). This is the basis for the classification network to work efficiently and accurately.

The single-stage system approach is aimed at developing a network system that performs both eye detection and eye status classification tasks at the same time. This method will be based on modifying the proposed eye detection network architecture with suitable solutions for human eye detection and classification as mentioned above. Besides, it is also necessary to build an appropriate eye detection dataset for training and evaluation. Even though, this work will spend more time and labor than the multi-stage method.

Appendix A

Publications

A.1 Journal

1. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Facemask Wearing Alert System Based on Simple Architecture with Low-Computing Devices, *IEEE Access*, 2022.
2. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Driver Behaviors Recognizer Based on Light-weight Architecture and Attention Mechanism, *IEEE Access*, 2022.
3. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, An Efficient Face Detector on a CPU Using Dual-Camera Sensors for Intelligent Surveillance Systems, *IEEE Sensors*, 2022.
4. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, A Fast CPU Real-time Facial Expression Detector using Sequential Attention Network for Human-robot Interaction, *IEEE Transactions on Industrial Informatics*, 2022.

A.2 Conference

1. Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo, Car Detector Based on YOLOv5 for Parking Management, *CITA 2023*, Danang, Vietnam, Jul 28, 2023 (Accepted).
2. Adri Priadana, Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Xuan-Thuy Vo, Kang-Hyun Jo, Age Group Recognizer based on Human Face Supporting

- Smart Digital Advertising Platforms, ISIE 2023, Helsinki-Espoo, Finland, June 21, 2022 (Accepted).
3. Xuan-Thuy Vo, Jehwan Choi, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Unifying Local and Global Fourier Features for Image Classification, ISIE 2023, Helsinki-Espoo, Finland, June 21, 2022 (Accepted).
 4. Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo, YOLO5PKLot: A Parking Lot Detection Network Based on Improved YOLOv5 for Smart Parking Management System, IW-FCV 2023, Yeosu, South Korea, Feb 20, 2023.
 5. Xuan-Thuy Vo, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Dynamic Circular Convolution for Image Classification, IW-FCV 2023, Yeosu, South Korea, Feb 20, 2023.
 6. Adri Priadana, Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Xuan-Thuy Vo, and Kang-Hyun Jo, Human Face Detector with Gender Identification by Split-based Inception Block and Regulated Attention Module, IW-FCV 2023, Yeosu, South Korea, Feb 20, 2023.
 7. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, An Efficient Multi-view Facial Expression Classifier Implementing on Edge Device, ACIIDS22, Ho Chi Minh, Viet Nam, Nov 28, 2022.
 8. Tran Tien Dat, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, Combination of Deep Learner Network with Transformer for 3D Human Pose Estimation, ICCAS 2022, Busan, Korea, Nov 27, 2022.
 9. Xuan-Thuy Vo, Tran Tien Dat, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Balancing Multiple Object Tracking Objectives based on Learned Weighting Factors, MAPR 2022, Phu Quoc Island, Vietnam, Oct 13, 2022.
 10. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, Tran Tien Dat, and Kang-Hyun Jo, Fire Warning Based on Convolutional Neural Network and Inception Mechanism, MAPR 2022, Phu Quoc Island, Vietnam, Oct 13, 2022.
 11. Tran Tien Dat, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, Efficient High-Resolution Network for Human Pose Estimation, IWIS 2022, University of Ulsan, Ulsan, South Korea, Aug 17, 2022.

12. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, Tran Tien Dat, and Kang-Hyun Jo, Robust Hand Detection Based on Convolutional Neural Network and Attention Module, IWIS 2022, University of Ulsan, Ulsan, South Korea, Aug 17, 2022.
13. Xuan-Thuy Vo, Tran Tien Dat, Duy-Linh Nguyen, and Kang-Hyun Jo, A Study on Efficient Multi-task Networks for Multiple Object Tracking, IWIS 2022, University of Ulsan, Ulsan, South Korea, Aug 17, 2022.
14. Muhamad Dwisnanto Putro, Adri Priadana, Duy-Linh Nguyen, and Kang-Hyun Jo, A Real-time Face Detector on CPU Using Efficient Transformer, IWIS 2022, University of Ulsan, Ulsan, South Korea, Aug 17, 2022.
15. Xuan-Thuy Vo, Tran Tien Dat, Duy-Linh Nguyen, and Kang-Hyun Jo, Multi-level Feature Reweighting and Fusion for Instance Segmentation, INDIN 2022, Perth, Australia, Jul 25, 2022.
16. Muhamad Dwisnanto Putro, Adri Priadana, Duy-Linh Nguyen, and Kang-Hyun Jo, A Faster Real-time Face Detector Support Smart Digital Advertising on Low-cost Computing Device, Advanced Intelligent Mechatronics (AIM), Sapporo, Hokkaido, Japan, Jul 11, 2022.
17. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Fast Person Detector with Efficient Multi-level Contextual Block for Supporting Assistive Robot, ICPS 2022, University of Warwick in Coventry, United Kingdom, May 24, 2022.
18. Tran Tien Dat, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, High-Resolution Network with Attention Module for Human Pose Estimation, ASCC 2022, Jeju, Korea, May 4, 2022.
19. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, A CPU-based Pedestrian Detector using Deep Learning for Intelligent Surveillance Systems, ICIT 2022, Shanghai, China, Mar 27, 2022.
20. Xuan-Thuy Vo, Van-Dung Hoang, Duy-Linh Nguyen, and Kang-Hyun Jo, Pedestrian Head Detection and Tracking via Global Vision Transformer, IW-FCV 2022, Hiroshima, Japan, Feb 21, 2022.

21. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and Kang-Hyun Jo, Convolutional Neural Network Design for Eye Detection under Low-illumination, IW-FCV 2022, Hiroshima, Japan, Feb 21, 2022.
22. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Light-weight Convolutional Neural Network for Distracted Driver Classification, IECON 2021, Toronto, Canada, Oct 13, 2021.
23. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Distracted Driver Recognizer with Simple and Efficient Convolutional Neural Network for Real-time System, ICCAS2021, Jeju, Korea, Oct 12, 2021.
24. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, A Fast Real-time Facial Expression Classifier Deep Learning-based for Human-robot Interaction, ICCAS2021, Jeju, Korea, Oct 12, 2021.
25. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, Efficient Face Detector Using Spatial Attention Module in Real-Time Application on an Edge Device, ICIC2021, Shenzhen, Guangdong, China, Aug 12, 2021.
26. Xuan-Thuy Vo, Tran Tien Dat, Duy-Linh Nguyen, and Kang-Hyun Jo, Regression-Aware Classification Feature for Pedestrian Detection and Tracking in Video Surveillance Systems, ICIC2021, Shenzhen, Guangdong, China, Aug 12, 2021.
27. Xuan-Thuy Vo, Tran Tien Dat, Duy-Linh Nguyen, and Kang-Hyun Jo, Dynamic Multi-Loss Weighting for Multiple People Tracking in Video Surveillance Systems, INDIN2021, Palma de Mallorca, Spain, Jul 21, 2021.
28. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and Kang-Hyun Jo, Triple Detector based on Feature Pyramid Network for License Plate Detection and Recognition System in Unusual Conditions, ISIE 2021, Miyako Messe, Kyoto, Japan, Jun 20, 2021.
29. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Eye State Recognizer Using Light-weight Architecture for Drowsiness Warning, ACIIDS 2021, Phuket, Thailand, Apr 7, 2021.
30. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, Real-time Multi-view Face Masks Detector on Edge Device For Supporting Service

- Robots in the COVID-19 Pandemic, ACIIDS 2021, Phuket, Thailand, Apr 7, 2021.
31. Xuan-Thuy Vo, Tran Tien Dat, Duy-Linh Nguyen, and Kang-Hyun Jo, Stair-step Feature Pyramid Networks for Object Detection, IW - FCV 2021, Daegu, South Korea, Feb 21, 2021.
 32. Tran Tien Dat, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, Efficient Spatial-Attention module for human pose estimation, IW - FCV 2021, Daegu, South Korea, Feb 21, 2021.
 33. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, SGC-Net: Spatial-Global Context Attention Network for Real-time Facial Expression Recognition, IWIS 2020, Ulsan, Korea, Dec 13, 2020.
 34. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Proposed Light-weight Convolutional Neural Networks for Real-time Hand Gesture Detector, IWIS 2020, Ulsan, Korea, Dec 13, 2020.
 35. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Human Eye Detector with Light-weight and Efficient Convolutional Neural Network, ICCCI 2020, Da Nang, Viet Nam, Nov 30, 2020.
 36. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Eyes Status Detector Based on Light-weight Convolutional Neural Networks supporting for Drowsiness Detection System, IECON 2020, Marina Bay Sands Expo and Convention Centre, Singapore, Oct 18, 2020.
 37. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, A Dual Attention Module for Real-time Facial Expression Recognition, IECON 2020, Marina Bay Sands Expo and Convention Centre, Singapore, Oct 18, 2020.
 38. Duy-Linh Nguyen, Muhamad Dwisnanto Putro, and Kang-Hyun Jo, Hand Detector based on Efficient and Lightweight Convolutional Neural Network, IC-CAS 2020, Busan, Korea, Oct 13, 2020.
 39. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, and Kang-Hyun Jo, Fast Eye Detector Using CPU Based Lightweight Convolutional Neural Network, IC-CAS 2020, Busan, Korea, Oct 13, 2020.

40. Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Kang-Hyun Jo, Lightweight Convolutional Neural Network for Real-Time Face Detector on CPU Supporting Interaction of Service Robot, HSI 2020, Tokyo, Japan, Jun 6, 2020.

Bibliography

- Ansari, Shahzeb, Fazel Naghdy, and Haiping Du (2022). "Human-Machine Shared Driving: Challenges and Future Directions". In: *IEEE Transactions on Intelligent Vehicles*, pp. 1–1. DOI: [10.1109/TIV.2022.3154426](https://doi.org/10.1109/TIV.2022.3154426).
- Araujo, Gabriel M et al. (2014). "Fast eye localization without a face model using inner product detectors". In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 1366–1370.
- Chen, Shuo and Chengjun Liu (2015). "Eye detection using discriminatory Haar features and a new efficient SVM". In: *Image and Vision Computing* 33, pp. 68–77.
- Chirra, Venkata, U. Srinivasulu Reddy, and Venkata KishoreKolli (Dec. 2019). "Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State". In: *Revue d'Intelligence Artificielle* 33, pp. 461–466. DOI: [10.18280/ria.330609](https://doi.org/10.18280/ria.330609).
- Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning*. Vol. 20. 3. Springer, pp. 273–297.
- Dalal, Navneet and Bill Triggs (2005). "Histograms of oriented gradients for human detection". In: *CVPR 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.* 1, pp. 886–893.
- Elgendy, Mohamed (2020). *Deep Learning for Vision Systems*. Manning.
- Everingham, M. et al. (Jan. 2015). "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1, pp. 98–136.
- Fawaz, Hassan Ismail et al. (2018). "Deep learning for time series classification: a review". In: *CoRR* abs/1809.04356. arXiv: [1809.04356](https://arxiv.org/abs/1809.04356). URL: <http://arxiv.org/abs/1809.04356>.
- Feng, Zhen-Hua et al. (2017). "Wing Loss for Robust Facial Landmark Localisation with Convolutional Neural Networks". In: *CoRR* abs/1711.06753. arXiv: [1711.06753](https://arxiv.org/abs/1711.06753). URL: <http://arxiv.org/abs/1711.06753>.

- Fürnkranz, Johannes (2010). "Decision Tree". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, pp. 263–267. ISBN: 978-0-387-30164-8. DOI: [10.1007/978-0-387-30164-8_204](https://doi.org/10.1007/978-0-387-30164-8_204). URL: https://doi.org/10.1007/978-0-387-30164-8_204.
- Fusek, R. (2018). "Pupil localization using geodesic distance". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11241 LNCS, pp. 433–444. DOI: [10.1007/978-3-030-03801-4_38](https://doi.org/10.1007/978-3-030-03801-4_38).
- Georghiades, A.S., P.N. Belhumeur, and D.J. Kriegman (2001). "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose". In: *IEEE Trans. Pattern Anal. Mach. Intelligence* 23.6, pp. 643–660.
- GI4E - Gaze Interaction for Everybody (n.d.). <http://www.unavarra.es/gi4e/databases?languageId=1>. Accessed: 2020-10-23.
- Girshick, Ross B. (2015). "Fast R-CNN". In: *CoRR* abs/1504.08083. arXiv: [1504.08083](https://arxiv.org/abs/1504.08083). URL: <http://arxiv.org/abs/1504.08083>.
- Girshick, Ross B. et al. (2013). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524. arXiv: [1311.2524](https://arxiv.org/abs/1311.2524). URL: <http://arxiv.org/abs/1311.2524>.
- Glenn Jocher, et al. (Oct. 2020). *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. Version v3.1. DOI: [10.5281/zenodo.4154370](https://doi.org/10.5281/zenodo.4154370). URL: <https://doi.org/10.5281/zenodo.4154370>.
- He, Kaiming et al. (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 346–361. DOI: [10.1007/978-3-319-10590-1_23](https://doi.org/10.1007/978-3-319-10590-1_23).
- (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- He, Kaiming et al. (2017). "Mask R-CNN". In: *CoRR* abs/1703.06870. arXiv: [1703.06870](https://arxiv.org/abs/1703.06870). URL: <http://arxiv.org/abs/1703.06870>.
- Howard, Andrew G et al. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv preprint arXiv:1704.04861*.
- Hu, Jie, Li Shen, and Gang Sun (2017). "Squeeze-and-Excitation Networks". In: *CoRR* abs/1709.01507. arXiv: [1709.01507](https://arxiv.org/abs/1709.01507). URL: <http://arxiv.org/abs/1709.01507>.
- Huang, Gao et al. (2017). "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

- Jahan, Israt et al. (2023). "4D: a real-time driver drowsiness detector using deep learning". en. In: *Electronics (Basel, Switzerland)* 12.1, e235–e235. ISSN: 2079-9292. DOI: [10.3390/electronics12010235](https://doi.org/10.3390/electronics12010235). URL: <http://dx.doi.org/10.3390/electronics12010235>.
- Khan, Tayab et al. (Mar. 2019). "Smart Real-Time Video Surveillance Platform for Drowsiness Detection Based on Eyelid Closure". In: *Wireless Communications and Mobile Computing* 2019, pp. 1–9. DOI: [10.1155/2019/2036818](https://doi.org/10.1155/2019/2036818).
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2017). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6, 84–90. ISSN: 0001-0782. DOI: [10.1145/3065386](https://doi.org/10.1145/3065386). URL: <https://doi.org/10.1145/3065386>.
- Leo, Marco et al. (2013). "Unsupervised approach for the accurate localization of the pupils in near-frontal facial images". In: *Journal of Electronic Imaging* 22.3, p. 033033.
- Lin, Tsung-Yi et al. (2017). "Focal Loss for Dense Object Detection". In: *CoRR abs/1708.02002*. arXiv: [1708.02002](https://arxiv.org/abs/1708.02002). URL: <http://arxiv.org/abs/1708.02002>.
- Liu, Wei et al. (2015). "SSD: Single Shot MultiBox Detector". In: *CoRR abs/1512.02325*. arXiv: [1512.02325](https://arxiv.org/abs/1512.02325). URL: <http://arxiv.org/abs/1512.02325>.
- Liu, Wei et al. (2016). "SSD: Single shot multibox detector". In: *European conference on computer vision*. Springer, pp. 21–37.
- Lowe, David G (2004). "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110.
- Markuš, Nenad et al. (2014). "Eye pupil localization with an ensemble of randomized trees". In: *Pattern recognition* 47.2, pp. 578–587.
- Misra, Diganta et al. (2020). "Rotate to Attend: Convolutional Triplet Attention Module". In: *CoRR abs/2010.03045*. arXiv: [2010.03045](https://arxiv.org/abs/2010.03045). URL: <https://arxiv.org/abs/2010.03045>.
- Mucherino, Antonio, Petraq J. Papajorgji, and Panos M. Pardalos (2009). "k-Nearest Neighbor Classification". In: *Data Mining in Agriculture*. New York, NY: Springer New York, pp. 83–106. ISBN: 978-0-387-88615-2. DOI: [10.1007/978-0-387-88615-2_4](https://doi.org/10.1007/978-0-387-88615-2_4). URL: https://doi.org/10.1007/978-0-387-88615-2_4.
- Park, Jongchan et al. (2018). "BAM: Bottleneck Attention Module". In: *CoRR abs/1807.06514*. arXiv: [1807.06514](https://arxiv.org/abs/1807.06514). URL: <http://arxiv.org/abs/1807.06514>.

- Qi, DeLong et al. (2021). "YOLO5Face: Why Reinventing a Face Detector". In: *CoRR* abs/2105.12931. arXiv: 2105.12931. URL: <https://arxiv.org/abs/2105.12931>.
- Ramzan, M. et al. (2019). "A Survey on State-of-the-Art Drowsiness Detection Techniques". In: *IEEE Access* 7, pp. 61904–61919. DOI: 10.1109/ACCESS.2019.2914373.
- Redmon, Joseph et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.
- Ren, Shaoqing et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497. arXiv: 1506.01497. URL: <http://arxiv.org/abs/1506.01497>.
- Ren, Shaoqing et al. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv: 1506.01497 [cs.CV].
- Road traffic injuries (n.d.). <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>. Accessed: Jan. 19, 2023. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>.
- Sathasivam, Saravananaraj et al. (2020). "Drowsiness Detection System using Eye Aspect Ratio Technique". In: *2020 IEEE Student Conference on Research and Development (SCoReD)*, pp. 448–452. DOI: 10.1109/SCoReD50371.2020.9251035.
- Saurav, Sumeet et al. (2022). "Real-time eye state recognition using dual convolutional neural network ensemble". In: *Journal of Real-Time Image Processing* 19.3, pp. 607–622.
- Shakeel, Muhammad Faique et al. (2019). "Detecting Driver Drowsiness in Real Time Through Deep Learning Based Object Detection". In: *Advances in Computational Intelligence*. Ed. by Ignacio Rojas, Gonzalo Joya, and Andreu Catala. Cham: Springer International Publishing, pp. 283–296. ISBN: 978-3-030-20521-8.
- Shang, Wenling et al. (2016). "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units". In: *CoRR* abs/1603.05201. arXiv: 1603.05201. URL: <http://arxiv.org/abs/1603.05201>.
- Sharma, Riti and Andreas Savakis (2015). "Lean histogram of oriented gradients features for effective eye detection". In: *Journal of Electronic Imaging* 24.6, p. 063007.
- Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

- Song, Fengyi et al. (Sept. 2014). "Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients". In: *Pattern Recognition* 47, 2825–2838. DOI: [10.1016/j.patcog.2014.03.024](https://doi.org/10.1016/j.patcog.2014.03.024).
- Świrski, Lech, Andreas Bulling, and Neil Dodgson (2012). "Robust real-time pupil tracking in highly off-axis images". In: *Proceedings of the symposium on eye tracking research and applications*, pp. 173–176.
- Szegedy, Christian et al. (2014). "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842. arXiv: [1409.4842](https://arxiv.org/abs/1409.4842). URL: <http://arxiv.org/abs/1409.4842>.
- Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.
- Tan, Mingxing and Quoc Le (2019). "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR, pp. 6105–6114.
- Tan, Mingxing, Ruoming Pang, and Quoc V. Le (2019). "EfficientDet: Scalable and Efficient Object Detection". In: *CoRR* abs/1911.09070. arXiv: [1911.09070](https://arxiv.org/abs/1911.09070). URL: <http://arxiv.org/abs/1911.09070>.
- The BioID Face Database* (n.d.). <https://www.bioid.com/facedb>. Accessed: 2020-10-23.
- Thomaz, Carlos and Gilson Giralaldi (June 2010). "A new ranking method for Principal Components Analysis and its application to face image analysis". In: *Image Vision Comput.* 28, pp. 902–913. DOI: [10.1016/j.imavis.2009.11.005](https://doi.org/10.1016/j.imavis.2009.11.005).
- Timm, Fabian and Erhardt Barth (2011). "Accurate eye centre localisation by means of gradients." In: *Visapp* 11, pp. 125–130.
- Tzutalin (2015). *LabelImg*. Free Software: MIT License. URL: <https://github.com/tzutalin/labelImg>.
- Valenti, Roberto and Theo Gevers (2011). "Accurate eye center location through invariant isocentric patterns". In: *IEEE transactions on pattern analysis and machine intelligence* 34.9, pp. 1785–1798.
- Viola, Paul and Michael Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE, pp. I-511.

- Wang, Robert J. et al. (2018). "Peele: A Real-Time Object Detection System on Mobile Devices". In: *CoRR* abs/1804.06882. arXiv: 1804.06882. URL: <http://arxiv.org/abs/1804.06882>.
- Wang, Shuai and Zhendong Su (2019). "Metamorphic Testing for Object Detection Systems". In: *CoRR* abs/1912.12162. arXiv: 1912.12162. URL: <http://arxiv.org/abs/1912.12162>.
- Wikipedia contributors (2023). *Convolutional neural network* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 11-April-2023]. URL: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1149194934.
- Woo, Sanghyun et al. (2018). "CBAM: Convolutional Block Attention Module". In: *CoRR* abs/1807.06521. arXiv: 1807.06521. URL: <http://arxiv.org/abs/1807.06521>.
- Yang, Shuo et al. (2015). "WIDER FACE: A Face Detection Benchmark". In: *CoRR* abs/1511.06523. arXiv: 1511.06523. URL: <http://arxiv.org/abs/1511.06523>.
- Zhang, Shifeng et al. (2017). "FaceBoxes: A CPU Real-time Face Detector with High Accuracy". In: *CoRR* abs/1708.05234. arXiv: 1708.05234. URL: <http://arxiv.org/abs/1708.05234>.
- Zou, Zhengxia et al. (2023). "Object Detection in 20 Years: A Survey". In: *Proceedings of the IEEE* 111.3, pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.