Eye Tracking System to Detect Driver Drowsiness using Deep Learning

Final Year Project Report

4th Year Student Name

Nguyen Viet Tung
HE130151
Hoang Manh
HE130294

Under the supervision of

MSc. Luong Trung Kien

Bachelor of Computer Science

Hoa Lac Campus – FPT University

Spring 2021

DECLARATION

Project Title	Eye Tracking System to Detect Driver Drowsiness using Deep Learning
Author	*Nguyen Viet Tung* and *Hoang Manh*
Student ID	HE130151 and HE130294
Supervisor	MSc. Luong Trung Kien

I declare that this thesis entitled *Eye Tracking System to Detect Driver Drowsiness using Deep Learning* is the result of my own work except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Nguyen Viet Tung and **Hoang Manh**

Department of Computer Science
Hoa Lac Campus – FPT University

Date: April 27, 2021

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my research supervisor, *MSc. Luong Trung Kien*, for giving me the opportunity to do research and providing invaluable guidance throughout this work. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work and study under his guidance.

I am greatly indebted to my honorable teachers of the Department of Computer Science at the Hoa Lac Campus – FPT University who taught me during the course of my study. Without any doubt, their teaching and guidance have completely transformed me to a person that I am today.

I am extremely thankful to my parents for their unconditional love, endless prayers, caring and immense sacrifices for educating and preparing me for my future. I would like to say thanks to my friends and relatives for their kind support and care.

Finally, I would like to thank all the people who have supported me to complete the project work directly or indirectly.

Nguyen Viet Tung and Hoang Manh
Hoa Lac Campus – FPT University
Date: April 27, 2021

ABSTRACT

An insufficient amount of sleep regimen can have an enormous impact on your quality of life. According to research, being subjected to stress at work, doing too much on the laptop, being on your smartphone, and experiencing problems with sleep deprivation. At the same time, driving can double the chances of you being tired behind the wheel. They are fatigued and drowsy while driving is a few of the reasons why there are more traffic accidents. Often, as a result of mental or physical exhaustion, people can fall asleep and face difficulties. This thesis discusses a method for determining whether a driver is sleepy behind the wheel and helps the person stop an accident. As a goal, one side effect of this initiative's overall goal is to cut traffic accidents. We want to boost drivers' alertness and make people's attention span longer. Masking drowsiness while driving might lead to frequent yawning and drooping of the eyelids, or getting progressively drowsier and start to fall asleep behind the wheel, might occur. We used the network-based face and eye-expansion feature extraction algorithm to identify the driver and locate his pupils. To calculate the percentage of eyelid closure over time, we use the driver's eye closing characteristic.

Keywords: Drowsiness; Driver Fatigue; Image Processing; Deep Learning; Transfer learning; Convolutional neural network; PERCLOS algorithm; Multi-task Cascaded Convolutional Networks;

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

According to the National Highway Traffic Safety Administration estimated that in 2017 has 91,000 police-reported crashes involved drowsy drivers (14). In the fact that the real world maybe has more accidents than the report. Over the year, with the number of vehicles increasing worldwide, it has become necessary daily. Many people are driving mentally and in a sleepy state and tend to move to the destination. Doing so can lead to drowsiness and uncontrolled driving, resulting in collisions with other vehicles and possibly death.

There are numerous non-driving-related causes to result in accidents, including road conditions, the weather, and the mechanical performance of a car. However, most of the accidents are happened due to the awareness and subjectivity of the driver. Hence, every year thousands of people die in road accidents, and the essential factor for this is the driver's drowsiness. The drowsiness behaviors related to fatigue nature are in any form, including eye closing, head-nodding, etc. Fortunately, the technology is developing, artificial intelligence and computer vision algorithms can become a solution to solve this problem. The research on the intelligent detection of drowsy driving can significantly reduce traffic jams and open the way to making autonomous cars in the future.

There have many challenges, methods, and research on driving drowsiness detection based on drivers' behavior caused by low-cost and diverse technology in recent years. However, the limitation of face detection and fatigue assessing in

complex and changing surroundings leads to the algorithm's performance down. Follow references (16), use PERCLOS (Percentage of Eyelid Closure Over the Pupil Over Time) algorithm to recognize the fatigued driver to obtain the driver's receiver and evaluate the driving status. The experiment shows that the highest performance of this solution in ideal conditions. It firstly assesses whether the driver's eyes are open or closed in the current frame according to the eyelid proportion covering the pupil. Then, PERCLOS is calculated by the number of closed eyes frames over a period of time. When the proportion of eyelid covering the pupil is over 80 percent, the eyes are identified as closed in the current frame. However, this solution does not consider the driver's differences, especially the differences in the size of eyes, which may cause misjudgment in practical applications. In some abnormal situations, the driver is wearing glasses, sunglasses, and in the night – where the condition is not in the system's idea. In these cases, a convolution neural network (CNN) can increase the system's accuracy to alert the driver effectively.

This paper presents the image processing method based on the Deep Learning approach combination with PERCLOS for driver drowsiness. Our practice focuses on the driver's eyes while riding a car as input and classifies the behaviors of sleepiness into two classes (normal and drowsy). Firstly, we use Multi-task Cascaded Convolutional Networks (MTCNN) (25) to detect the face from the real-time camera. After that, we calculated all of the eyes' landmark coordinates to identify eyes' status (open or closed). Finally, we are using a convolutional neural network (CNN) model to tracking the driver's eyes. With the model, the actions can recognize by following the driver's face's eyes to help the human have a safe ride. The system can easy to deploy on the embedded board while driving and keep the accuracy. Since the person feels drowsy in real-time, the model instantly can detect and alert the person to have some solutions to handle sleepiness.

1.2 Outline

Chapter 1: It is a general introduction about drowsiness detection, the scope of this thesis.

Chapter 2: We provide a detailed overview of the technical and neural networks. In

this chapter, we present the problem's approach, the architect of the neural network we use, and some methods in our experiment.

Chapter 3: From the technology we have outlined in chapter 2, in this chapter, we will give the technologies we used in our tracking eye system to tell whether the driver is sleepy or not and provide a warning for the driver and minimize unwanted traffic accidents.

Chapter 4: In this chapter, we provide the results from the training model, testing the above system in some cases we have listed for testing.

Chapter 5: It includes the results of testing our systems and some ways to extend the system in the future.

1.3 Related Works

With the advent of new technologies, a lot of quick, accurate machine learning models are coming to the fore, which lessens the workload and, as well as guaranteeing high recognition rates of accuracy. The use of strong artificial intelligence techniques such as machine learning and deep learning is recently widely accepted as a solution to exhaustion detection of human micro-sleep or state of mind. We will cover how we implemented deep learning models to reduce overfitting and how they can help in general to identify sleepiness when driving in this segment.

Chapter 2

BACKGROUND

2.1 Conventional Approaches for Drowsiness Detection

The driving trend may be determined by measuring the rotation of the steering wheel or the distance from the lane or lateral direction. Micro-adjustment of the steering wheel is essential while driving to keep the car in the street. Following (9) achieved an accuracy of 86% in drowsiness detection based on correlations between micro-adjustments and drowsiness.

The second class of techniques incorporates data from physiological sensors such as EEG, ECG, and EOG data. EEG impulses provide information about brain function. Three key cues for calculating the driver's sleepiness in EEG are alpha, delta, and theta signals. When the driver is drowsy, the delta and theta signals jump up, and the alpha signal increases marginally. In paper (12), this technique gives the best accuracy among all the three methods (more than 90%). Although this solution has advantages, one major disadvantage is that may bother the driver is having sensors installed in the vehicle. However, non-invasive bio-signals do not function.

The above is focused on retrieving facial features using Computer Vision, where patterns such as eye closing, shifting of the head, gaze, or facial expression have been used. The references (4) used the distance between eyelids to measure drowsiness of 3 levels. The distinction was made based on the number of blinks per minute, on the premise that the count increases as the individual gets drowsier.

2.2 Neural Network

Before we begin discussing CNN, we can tell you about how all the other popular neural networks function(18). Neural Network (NN) is a model of programming that simulates the functioning of human neural networks (1). In combination with deep learning (DL), neural networks are a versatile method that can better achieve numerous complex problems, including image recognition (8), language recognition (22), or processing (17). In several fields of science and technology, artificial intelligence is today becoming a common subject. Apple's Siri-virtual assistant, Tesla's self-driven automotive, and Netflix's movie recommendation system are some of the standing applications in the Google Translate multi-language localization system.

The neural network history started with developing a neural network computing model by Warren McCulloch and Walter Pitts in 1943 based on algorithms called threshold logic (3). For instance, the dog can differentiate among family members and foreigners, or the kid may differ between species. Things look very simple but very hard to do with a computer. The response is that the brain has several neurons connected. Neural is a neuronal adjective, network graph only, so the neural network (NN) is a computer device influenced by neuronal activation in the nervous system.

The neuron is the nervous system's essential component and the most significant component of the brain. Our brain comprises around 10 million neurons, and each neuron connects to 10,000 other neurons. Each neuron has a somatic core body, dendrites input, and axon output signals attached to other neurons. The dendrites obtain the input data, and the output data are sent to other dendrites. The signal passes through the axon through the dendrites of other neurons if the electrical pulses are adequate to transform the nucleus into a neuron. So any neuron must determine whether this neuron should be activated or not.

In summary, there are functional approximation models for brains and empirical models that can help in all areas of speculation, particularly in drawing upon what we know about the brain's observations. The design and learning mechanism are explained in this portion if you're interested in learning more about neural networks, which components make up neural networks.

2.2.1 Neural network components

Units. Neural networks consist of units, which take inspiration from biological neurons. Each unit, also known as an artificial neuron, takes one or more data inputs and produces output, shipped off numerous units. The input of a unit is external data or output of one or more units.

Besides units, the network consists of connections that are defined as the relation between two units. These relations are measured by weights which are the strength of the connections. Weights influence the measure of the impact an adjustment in the input will have upon the output. Along with weights, biases, which are constants, are added into the input of the following layer. Biases speak for how isolated the predictions are from the desired values, and they also ensure that the input of units can be activated when they are equal to zero. To produce the output of a unit, first, we sum the weighted inputs and add bias to it; the result of this process is called activation $\boldsymbol{a}$. $\boldsymbol{a}$ is given by:

$$a = W^T x + b \tag{2.1}$$

In the final step, activations will be passed through the activation function. The value we receive after this process is the output of the unit. (Figure 2.1)

Activation Function This is a sub-unit of a neural network, which is a statistical mapping that defines the contribution of units. The activation function is often referred to as a transforming function because it converts the data. For instance, in binary sorting problems, the input is passed into an activation function to obtain output in the range 0 to 1.

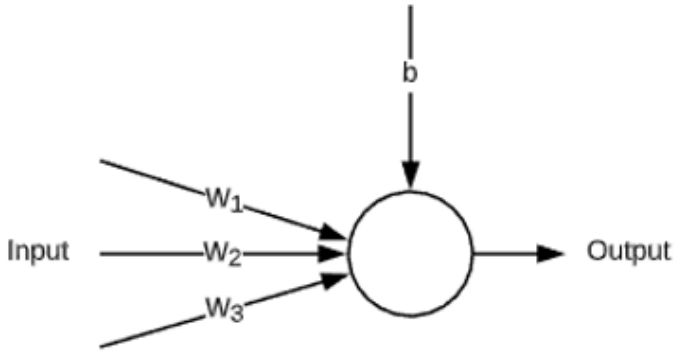

Figure 2.1: A unit in the Neural Network

The activation function may be classified into two types: linear and non-linear. The mechanism of linear activation, or the function of identity activation, is defined as f(x) = x. The linear activation function returns to output in no certain range (-∞, ∞), that does not help anything in transform the input. In practical, the non-linear activation function is used to transform the input. In fact, the input is transformed using a non-linear activation mechanism. This capstone project introduces two of the most common non-linear activation mechanisms, sigmoid functions, and ReLU.

ReLU The most common activation function for CNNs and also the one used in this thesis is the ReLu follow equation (2.2). It outputs x when x is positive and outputs 0 otherwise (Figure 2.2).

$$A(x) = max(0, x) \tag{2.2}$$

ReLu is less computationally expensive than some other common activation functions like tanh and Sigmoid because the mathematical operation is simpler and the activation is sparser. Since the function outputs 0 when x 0, there is a considerable chance that a given unit does not activate at all. Sparsity also means more concise models with more predictive power and less noise or overfitting. In a sparse network, neurons are more likely to process meaningful information. For example, a neuron which can identify human faces should not be activated if the image is

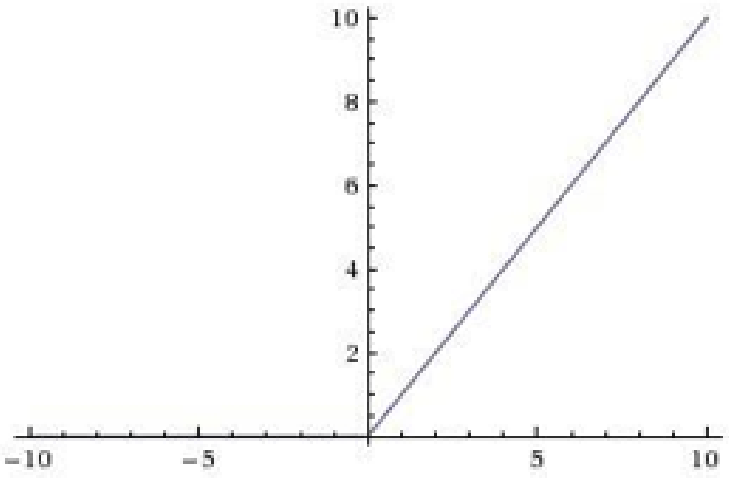

Figure 2.2: The ReLu function

actually about a building.

Sigmoid is frequently used to refer to the logistic equation, alternatively known as the logistic sigmoid function. Since all sigmoid functions map the whole number line to a finite range, such as between 0 and 1, or -1 and 1, one use is to transform a real value to one that can be viewed as a probability.2.3

The logistic function is one of the most frequently used sigmoid functions; it maps every actual value to a set (0, 1). Take note of the distinctive S-shape that gives sigmoid functions their reputation (from the Greek letter sigma). Since they can be used as an initiation function in an artificial neural network, sigmoid functions have gained popularity in deep learning. They were motivated by biological neural networks' activation capacity.

Sigmoid functions are often useful in a variety of machine learning applications that include the conversion of a specific number to a likelihood. A sigmoid feature applied as the final layer of a machine learning algorithm may be used to transform the model's performance to a likelihood score, which is more manageable and interpretable.

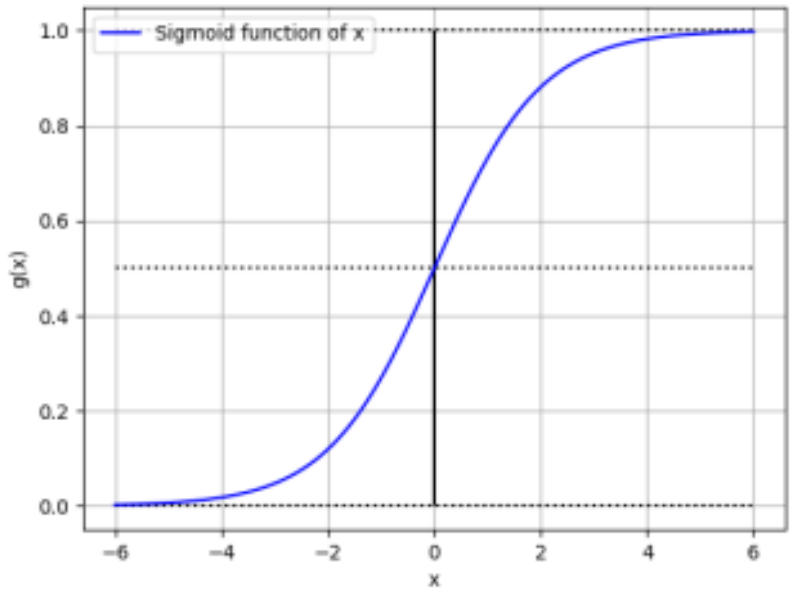

Figure 2.3: The Sigmoid function

Sigmoid functions are an important part of a logistic regression model. Logistic regression is a modification of linear regression for two-class classification. It converts one or more real-valued inputs into a probability, such as the probability that a customer will purchase a product. The final stage of a logistic regression model is often set to the logistic function, which allows the model to output a probability.

Layers units in a neural network are organized into multiple blocks, which are called layers. In neural networks, units in the same layers do not connect to each other; they only connect to units in the previous layers and units in the next layers.

There are three types of layers in a neural network (Figure 2.4). Each network has only one input layer and one output layer. The input layer is the first layer of a network, while the output layer is the last layer of a network. Besides the input layer and output layer, a multi-layer neural network can have zero or more hidden layers. Layers in the network are arranged in the order of input layer, hidden layers, and output layer. The number of layers in a network is denoted by L, which is calculated by summing the number of hidden layers and the number of output

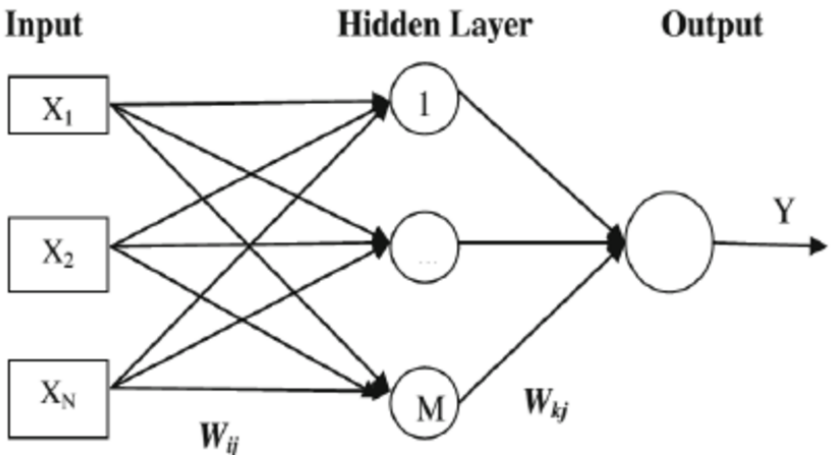

Figure 2.4: Layers of a neural network

layers. The first layer, or the input layer, is denoted as the 0^{th} layer.

If there is more than one layer in a network, it is called a multi-layer neural network. The fully connection between 2 layers is that every neuron in a layer connects to every neuron in another layer. In a feedforward neural network, except the input layer, layers take input from the output of the immediately preceding layer. Each unit in the layer sends its output to the following layer, and it will be transformed by operation and become the input of the following layer. The output of the output layers is also the output of the models, which are predicted values. Outputs of the hidden layers are not shown. That is the reason why they are called hidden.

Learning rate is hyperparameter specifies the amount by which the algorithm is changed in response to the expected error each time the model weights are adjusted. Choosing the learning rate is difficult since a value too small may result in a prolonged training phase that can get stuck, while a value too high may result in acquiring an inefficient range of weights too quickly or in an unpredictable training process.

When configuring the neural network, the learning rate can be the most critical hyperparameter. As a result, it is critical to understand how to investigate the impact of the learning rate on model success and to develop an understanding of the learning rate's dynamics on model behavior.

2.3 Convolutional Neural Network

Artificial Intelligence (AI) has grown through various scientific and technical disciplines over the last few decades. It will develop templates for itself by using the input data and then self-modify based on knowledge. Convolutional Neural Networks (CNN) are one of the most widely used architectures for image-based deep learning (2). Whereas traditional machine learning relevant features have to be extracted manually, deep learning uses raw images as input to understand specific features. CNNs consist of an input and output layer and several hidden layers between the input and output.

Convolutional Neural Networks (CNNs) is a Deep Learning algorithm that can take an input image and generate an output image, allocate significance (learnable weights and biases) to various aspects/objects in the image, and distinguish between them (7). A ConvNet needs far less pre-processing than other classification algorithms. Although primitive methods include hand-engineering of filters, ConvNets may acquire these filters/characteristics with sufficient preparation.

A ConvNet's architecture is similar to the connectivity structure of Neurons in the Human Brain and was influenced by the Visual Cortex's organization. Individual neurons react to a stimulus only within a small area of the visual field referred to as the Receptive Field. A set of such domains will converge to fully fill the pictorial space. While the approach may have an average precision score when performing class prediction on clear binary images, it would have little or no accuracy when performing class prediction on complex images with pixel dependencies.

Through the application of appropriate filters, a ConvNet is capable of successfully capturing the spatial and temporal dependencies in a picture. Owing to the reduced number of parameters and reusability of weights, the architecture performs a closer match to the image dataset. In other terms, the network may be programmed to recognize the image's sophistication.

In conclusion, rather than seeing convolutional neural networks as cerebrum job

constructs, it is preferable to think about them as practical inference devices designed to accomplish factual hypotheses, building on a few examples from what we know about the brain instances. This section introduces an architecture convolutional neural network, including components, learning process, and neural network for drowsiness detection.

2.3.1 Convolutional neural network components

The convolution layer (Conv layer) is where all the computational power is concentrated in a CNN. Due to being usually fixed to being stationary, images are stable, which is quite significant. When we expand this phrase, we are saying that each element is entirely and identically formed. A function learned in one part of the world bears a resemblance to a similar pattern in another in a preeminent way explicitly. Take a small section of the image and then particularly expand it to take on the characteristics of a broad selection of the image (Input). When in motion, the particles are mostly spaced out and positioned to form a generally single trajectory (Output), which is significant at every point in their path. That essentially is a method of converting the kind of smaller portion of the picture to the more extensive section of the same image (Kernel). To calculate the new values and type of expanding the search by the backdrop technique.

Convolutional Neural Networks are similar to multilayer perceptrons (and can be said to be comprised of individual MLPs, but this comparison is somewhat unclear) in that they include a single input layer, a single output layer, and a collection of – at least one – unknown layer(s) in between. CNN is made up of several 'components.' A part in a CNN is equivalent to at least one sheet. It is composed of three components: Layers with convolution, layers with pooling, and layers with full connectivity.

2.3.1.1 Convolutional layers

Convolution is a specialized method of a linear process that is used for function extraction. It is performed by applying a small array of numbers called a kernel over the data, which is an array of numbers referred to as tensor (24). At each place of the tensor, and element-wise product for each element of the kernel and the input tensor

is computed and summed to obtain the output value in the corresponding direction of the output tensor, referred to as a feature map. This procedure is repeated for an arbitrary number of kernels to generate a random number of feature maps that represent the input tensors' distinct characteristics. Hence, separate kernels may be thought of as different feature extractors. The size and number of kernels are two critical hyperparameters that characterize the convolution process. The former is usually 3x3, but maybe 5x5 or 7x7 at times. The above is entirely arbitrarily chosen and defines the depth of the output feature maps. (Figure 2.5)

An image to be categorified is provided to the input layer, and output is that the expected class label computed exploitation extracted options from image. A personal somatic cell within the next layer is connected to some neurons within the previous layer. This native correlation is termed as receptive field. The native options from the input image are extracted victimization receptive fields. The receptive field of a somatic cell associated to the specific region in the previous layer forms a weight vector that remains equal in any respect points on the plane, wherever the plane refers to the neurons within the next layer. Because the neurons in the plane share the same weights, so the similar options occurring at totally different locations in the input file are often detected. This has been represented in the figure to show below.

2.3.1.2 Pooling layer

The first secret sauce that has contributed significantly to CNN's effectiveness is pooling. Pooling is a vector to the scalar transformation that, like convolutions, operates on each local region of an image. They, though, lack filters and do not compute dot products with the local field, in contrast to convolutions. Other than that, they either apply the area's average of the pixels (Average Pooling) or choose the pixel with the maximum strength and delete the remainder (Max Pooling).(Figure 2.6)

The very idea of pooling can seem counter-productive as it leads to loss of information. However, it has proven to be very effective in practice because it makes covnets invariant to variations in the presentation of an image. It also reduces the effects of background noise. Max Pooling has worked best in recent years, it

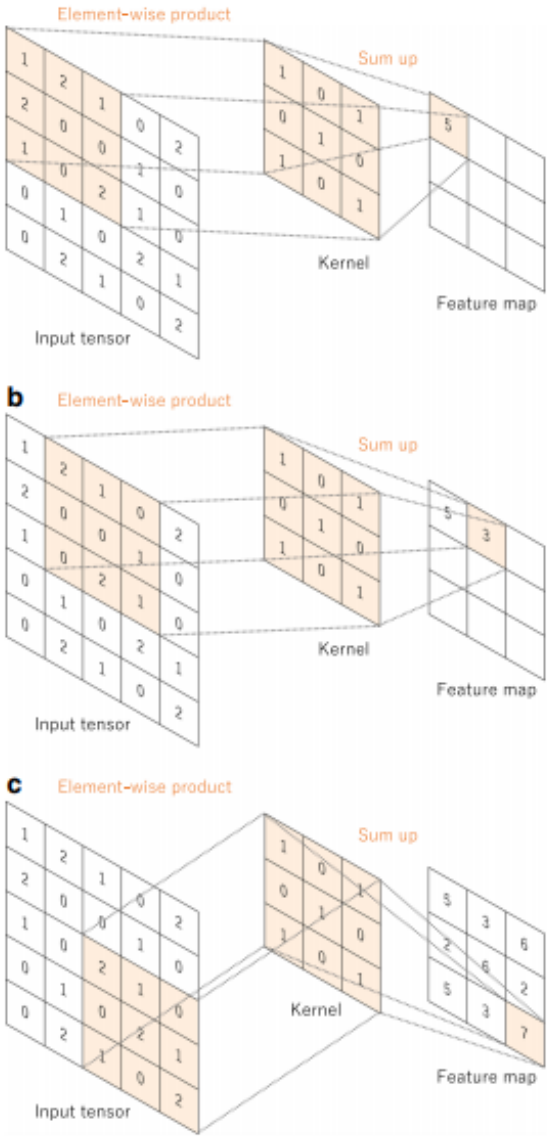

Figure 2.5: The convolution step: Image with the kernal 3x3 and output feature maps

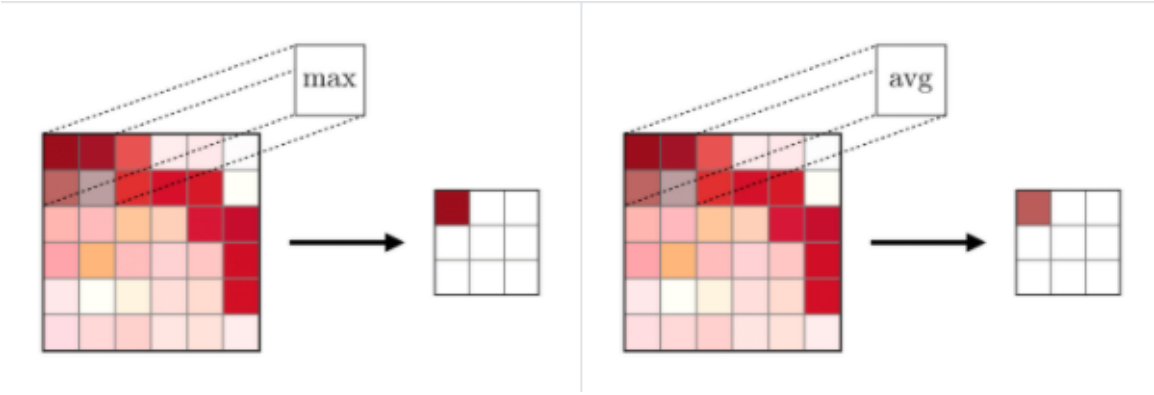

Figure 2.6: Max pooling: Each pooling operation selects the maximum value of the current view (Left); Average pooling: Each pooling operation averages the values of the current view (Right)

is based on the idea that the maximum pixel in a region represents the essential feature in that region. Often images of objects we wish to classify could contain several other things. For example, a cat appearing somewhere in the picture of a car could mislead the classifier. Pooling helps to alleviate the effects of this and makes covnets generalize better.

It also greatly reduces the computational cost of the covnet. Generally, the size of images at each layer in the network is directly proportional to the computational cost (flops) of each layer. Pooling reduces the dimensions of the image as the layers get deeper. Hence, it helps prevent an explosion of the number of flops a network requires. Stride convolutions are sometimes used as an alternative to pooling.

2.3.1.3 Fully Connected layer

The last layer of the CNN model in the image classification problem is the fully connected layer. This layer can convert the feature matrix in the previous layer into a vector containing the probabilities of the objects that need to be predicted.

The process of training the CNN model for the image classification problem is similar to training other models. An error function is required to calculate the error

between the model's prediction and the correct label and use the backpropagation algorithm for the weight update process.

2.3.2 Convolutional neural network architecture

Each input neuron is connected to each output neuron in a conventional feedforward neural network in the following layer. This is often referred to as a fully linked layer or affine layer. That is not how CNNs operate. Other than that, we calculate the output using convolutions over the input layer. As a consequence, each area of the input is bound to a neuron in the output. Each layer adds various filters, usually hundreds or thousands such as those seen above, and then blends their production. There is also something called pooling (subsampling) layers, which I will discuss later. A CNN dynamically learns the values of its filters during the testing process depending on the mission at hand. For instance, in Image Classification, a CNN can learn to detect edges from raw pixels in the first layer, then use such advantages to detect basic shapes in the second layer, and finally use these shapes to prevent higher-level features facial shapes in subsequent layers. The final layer is a classifier that makes use of these top-level functions. Follow the Figure 2.7 we can see the basic architecture of CNN to detect car in image.

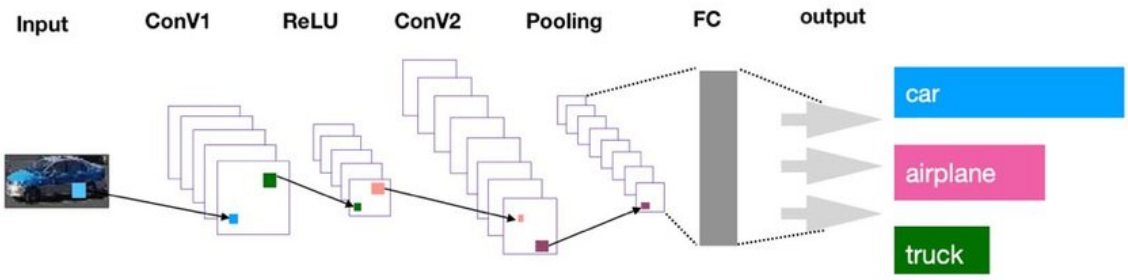

Figure 2.7: Basic architecture of Convolutional Neural Network.

2.3.3 Training the convolutional neural network

Adjusting the weights of individual neurons to obtain the correct characteristics from photos is one of the most difficult aspects of creating CNNs. Adjusting these weights is referred to as "training" the neural network.

The CNN begins with arbitrary weights. The developers feed the neural network a huge dataset of photos annotated with their associated groups during processing (cat, dog, horse, etc.). Each picture is processed using random values and then compared to the right mark using the ConvNet. If the network's performance does not fit the label at the start of the training phase, it makes a slight change to the weights of its neurons such that the next time it sees the same picture, its output is a little closer to the correct response.

Backpropagation is used to make the corrections (or backprop). Backpropagation essentially optimizes the tuning mechanism by allowing the network to choose which units to modify rather than requiring arbitrary adjustments.

Each iteration of the entire training dataset is referred to as an "epoch." Throughout the preparation, the ConvNet passes through many epochs, changing the weights in tiny increments. Every epoch improves the neural network's ability to distinguish the training videos. As the CNN progresses, the weight changes it produces become increasingly lower. At any stage, the network "converges," which ensures it becomes the best it can be.

After educating the CNN, the developers validate its accuracy using a reference dataset. The evaluation dataset consists of a collection of labeled photos that were not used in the training sample. Each picture is fed into the ConvNet, and the output is compared to the image's real name. Essentially, the research dataset assesses the neural network's ability to identify previously unseen images.

If a CNN performs well on training data but poorly on evaluation data, it is considered "overfitted." This usually occurs when the training data is insufficiently varied or when the ConvNet undergoes an excessive number of epochs on the training dataset.

Convolutional Neural Networks' performance is mainly attributed to the availability of massive image datasets built over the last decade. The competition listed at the beginning of this article is named after a dataset of over 14 million labeled photographs. Additional datasets are more specialized, such as the MNIST, which

contains 70,000 photos with handwritten digits.

However, there is no requirement to train per convolutional neural network on millions of photos. Sometimes, you may take a pre-trained model, such as the AlexNet or Microsoft's ResNet, and finetune it for more advanced users. This is referred to as transfer learning, and it involves retraining a learned neural network on a smaller number of new instances.

2.4 Multi-task Cascaded Convolutional Networks (MTCNN)

While there are many face detection methods (20)(19)(10)(26)(25), but MTCNN (Multitask Convolutional Neutral Networks) is adopted in this work for two reasons. On one hand, it achieves a high detection accuracy, and on the other hand, FaceNet model has already provided MTCNN interface to detect faces.

Multi-task Cascaded Convolutional Networks (MTCNN) is a deep learning algorithm for face recognition built on a cascaded Convolutional Neural Network with multiple tasks (CNN). It increases its efficiency by using the intrinsic link between identification and alignment. This paper's approach leverages a cascaded architecture with three levels of carefully constructed deep convolutional networks to forecast face and landmark locations in a coarse-to-fine (25). Figure 2.8 shows a photo of model of MTCNN

2.4.1 Proposal network (P-NET)

The P-net is a convolutional neural network in its entirety (FCN). The forwarding propagation function map is a 32-dimensional feature vector at each point. It is used to decide whether or not 12 x 12 grid cells have faces. If a grid cell includes a human face, the human face's Bounding Box is regressed to obtain the Bounding Box corresponding to the region in the original picture. A Non-maximum suppression (NMS) stage retains the Bounding Box with the highest ranking, and all other Bounding Boxes with an overly broad overlapping region are excluded.

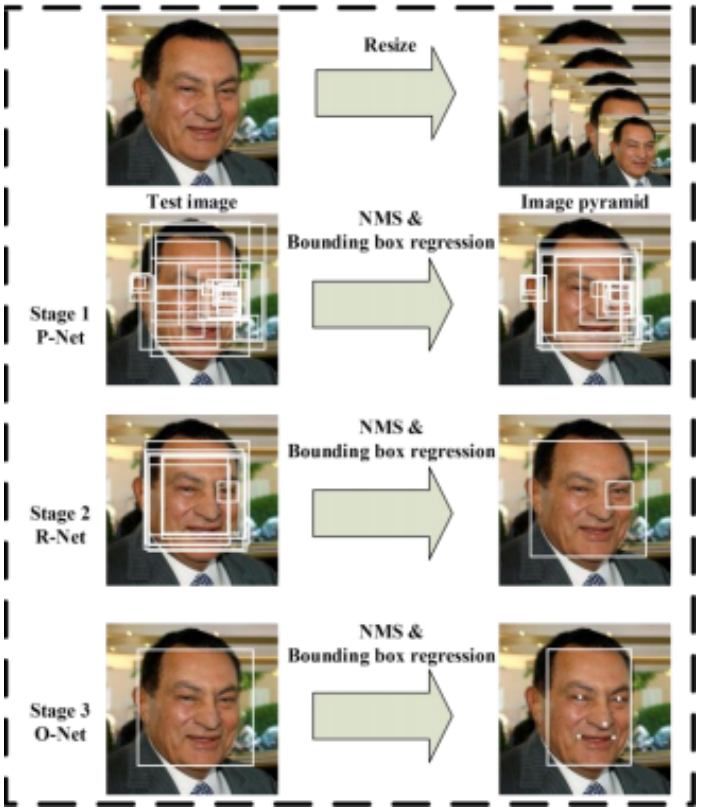

Figure 2.8: MTCNN: steps the cascaded architecture takes to ensure best performance in human face detection and bounding box regression.

2.4.2 Refine network (R-NET)

R-Net is a specific CNN point. As with the previous stage (O-Net), the 24x24 and corresponding Bounding Box area are enlarged to 48x48. It is then passed to the R-Net level, which is tasked with detecting Bounding Boxes and facial landmarks with the highest detection confidence.

2.4.3 Output network (O-NET)

O-NET is intended to improve precision. It is a straightforward CNN; the Bounding Box created by the P-Net move can or may not contain a human face. This package, along with the 12x12 input, is first up-scaled to 24x24 using a bilinear interpolation technique and then used as the input to the O-Net to evaluate the presence of a human face. When a human face is contained, the Bounding Box is regressed and the NMS level.

2.4.4 Architecture of MTCNN

The layer architectures are used in each stage of the cascaded MTCNN model. Each phase produces the same class of results using a different set of Conv filters and a different number of layers. Three types of outputs are available. Two neurons are used to generate the first series of outputs, which is the face classification score. One is used to indicate the presence of a face, while the other is used to indicate the ranking. Another section of the output is the bounding box regression, in which four neurons represent the bounding box's upper left and lower right corners as d_{x_1}, d_{y_1}, d_{x_2}, and d_{y_2} respectively. Localization of facial landmarks involves regressing five points: the left eye, the right eye, the nose, the left mouth corner, and the right mouth corner.

The three networks would use the landmark locations as supervised cues to direct the network's learning throughout the training process. However, during the prediction process, P-Net and R-Net perform only face detection and do not produce landmark locations due to inaccuracy in these areas. The O-Net is used to determine the landmark location. The outputs of the bounding box and landmarks synchronization are normalized about the data.

As mentioned above, there are three tasks that Multi-task Cascaded Convolutional Networks (MTCNN) archives. Precisely, mask recognition, bounding box regression, and localization of facial landmarks. As a result, the algorithm's loss function also has three parts. Due to space constraints, the following is a summary of the main points; readers seeking further information are directed to (21). Figure 2.9 illustrates the model's design for face detection and landmark extraction in detail.

2.5 The percentage of eyelid closure over the pupil (PERCLOS)

Wierwille et al. (13) developed PERCLOS as a drowsiness metric. PERCLOSURE can be described as the percentage of time that the eyelids are closed over the pupil by at least 80%. PERCLOS values can be calculated using continuous video sequences of eye pictures (6). Until calculating the correct PERCLOS amount,

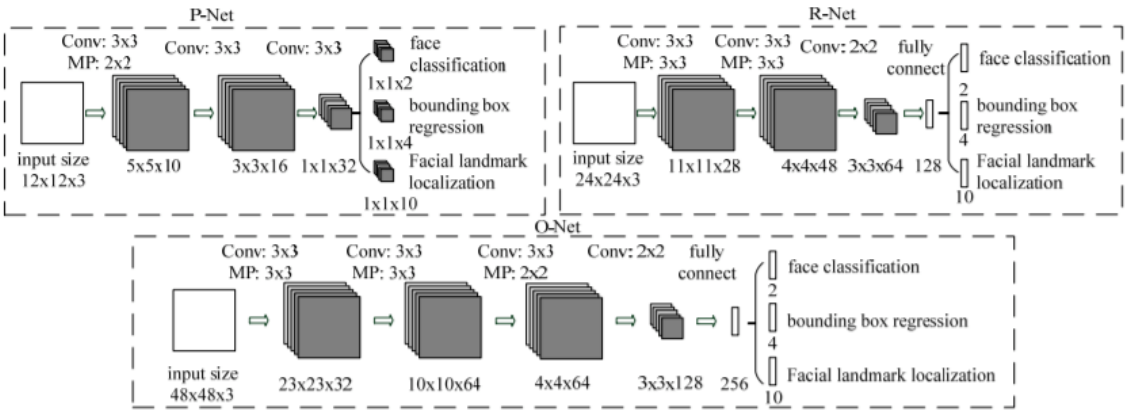

Figure 2.9: Architecture of MTCNN

multiple measures must be completed. Face identification comes first, accompanied by eye detection and description of eye states. Finally, the following formula is used to determine the PERCLOS value.

$$P = \frac{E_c}{E_t} \tag{2.3}$$

In the above equation (2.3), P is the PERCLOS value , E_c is the number of closed eye count over a predefined interval, while E_t the total eye count in the same given interval. E.g., suppose 100 frames of facial images are recorded and 20 such cases are found where the eyes are closed (at least 80% spatially). In that case, the PERCLOS value is 20%. The eyes are categorized as open or closed during training based on 80% spatial closure.

Given the detrimental effect of driver drowsiness on driving protection, there has been considerable interest in designing a device capable of monitoring and quantifying driver drowsiness and providing real-time warnings to the driver and controlling the car's performance. There are various drowsy driver tracking systems on the market, but almost all of them depend on a single indicator of driver drowsiness (e.g., eye closures). One disadvantage of utilizing a single predictor of drowsiness is that they are vulnerable to intermittent data gaps caused by sensor loss or use beyond the sensor's operating envelope.

Slow eye closing is one of the measures seen with some drowsiness monitors

(i.e., PERCLOS). However, tests of certain specific technologies have shown periodic data loss under various circumstances, including whether the driver is wearing eyewear (e.g., glasses, sunglasses). The driver's eyes are not inside the system's field of view due to standard visual scanning patterns (e.g., mirror checks), or the driver is doing a secondary task (e.g., looking down at the speedometer) that causes the driver's eyes to be outside the system's field of view.

Chapter 3

METHODS

3.1 Architecture

A general solution has two significant steps: the identification of drowsiness and the elaboration of a strategy. Datasets and preprocessing should be finished, then refer to the flow diagram in Fig 3.1. Drowsiness may be detected through facial feature recognition and facial mapping (23) . However, there is a specific and different process of finding standard facial features to joint landmarks and checking that they are in position. Using Multi-Task Cascade Classifier (MTCCN) (25) is usually comes with both the added to face detection and face alignment to obtain both accuracy and time. Hence they are given priority in this task. Exploiting the complicated structure can get high performance in facial recognition and segmentation. Two positions on left eye coordinates and the right-eye landmarks are often determined as a result of face recognition and orientation. Having the left-eye coordinates, right-eye coordinates we create bounding box and expansion measures completed is necessary to accomplish the data's label, which will produce bounding box and cropping landmark of left-right eye's status outcomes. Afterward, the previously mentioned method, the CNN model, loads data with the label to training and returns it to the summary model to detect drowsiness. Next step is application model to detect drowsiness while driving. Following Fig 3.2 we can observed all our system. In this step, applying the results we trained for in the first phase, which utilizes the PERCLOS algorithm (13) to calculate ratio frame of the camera and warn the driver if the driver is tired, and then repeat this step 2 more often before the ratio between the two sensors deviates significantly from their initial value.

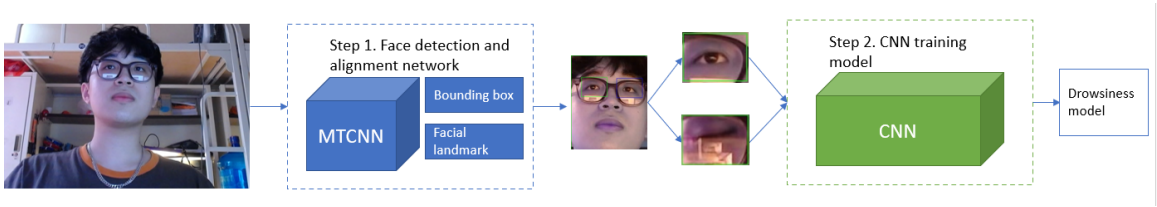

Figure 3.1: Overall step of Datasets and Preprocessing: Step 1 consists of face and landmark to get close, open status of eyes and step 2 consists of drowsiness model

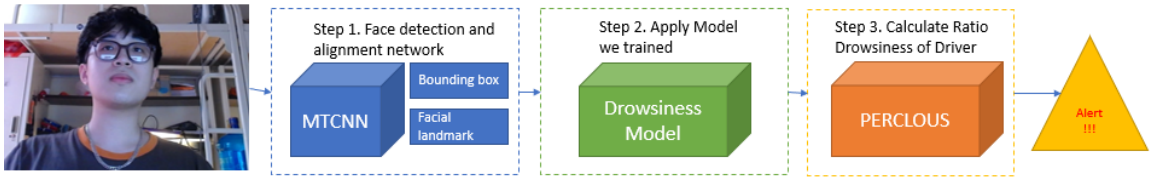

Figure 3.2: Overall step Detect Drowsiness Driver: Step 1 detect face and consists of face and landmark of eyes and step apply drowsiness model Step 3 Using PERCLOS to alert drowsiness

3.2 Datasets and Preprocessing

3.2.1 Datasets

The data we collect is the raw and the machine can not understand fully the meaning of this. Doing so many as converting picture, image, and video data to 1s, and 0s into information is limited to machines. Because our image recognition model would just need photos to be more trained after this, we can only have images and videos in our slideshow before it gets finished. In certain contexts, a dataset may be interpreted as a list of individual data items called documents, points, patterns, incidents, cases, or tests, but it can also be described as interconnected records, points, or patterns. The objects can be represented by a variety of attributes, such as the weight or the period or occurrence of an incident Features can also are named lengths, characteristics, factors, or properties depending on whether you choose to say they are.

The Deep Learning model developed here is trained on videos obtained from the FPT University student we collected and crawl data from the website. These videos are preprocessed to create images for this project.

1) FPT University Student Datasets: This dataset contains 30 subjects including from different human, both genders, wearing glass and bare eye as Fig 3.3. We collect the videos data of FPT Univesity Student with the status opening and closing eye. For each frames, we crop eye image as Figure for the model train.

Figure 3.3: Some samples of FPT University

2) Crawl Datasets in the internet: Face dataset is the common data in the internet. The internet is the open source and data for every field, it has many dataset for us to training data. We clone the data and make it like input for the model to training. From the MRL Eye Dataset (5), the large-scale dataset of human eye images. This dataset contains infrared images in low and high resolution, all captured in various lightning conditions and by different devices. The dataset is suitable for testing several features or trainable classifiers. In order to simplify the comparison of algorithms, the images are divided into several categories, which also makes them suitable for training and testing classifiers (Figure 3.4).

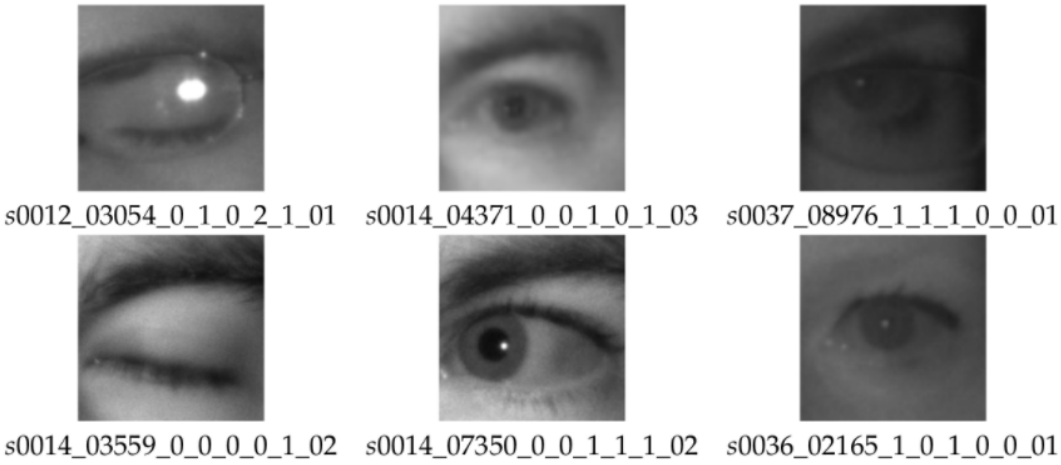

Figure 3.4: MRL Eye Dataset

3.2.2 Preprocessing

After the inception of convolutional neural networks (CNNs), a family of deep machine learning algorithms, computers have been created that can conduct facial recognition and verification tasks as efficiently as humans (15). The ability of intelligent machines to successfully perform recognition tasks depends on the CNN architecture and the format of the input data. For example, the dataset size may be inadequate for training a neural network. The data distribution may vary, deterioration due to noise or grey-level resolution may be an issue, and the color space may be different. Additionally, there might be instances of intra-person disparity due to variations in posture and illumination. Additionally, inter-person similarities can occur when the features of members of distinct classes bear a striking resemblance. Since the latter is more common in face space than in object space, it is not covered in this analysis.

Typically, the size of the dataset used to train a CNN is critical for deciphering the complex trends found in the results. The literature often uses data augmentation techniques such as translation, rotating, scaling, and reflection to solve the limited sample size problem. Another often-used technique is data normalization, which is used to minimize variance in data distribution. It works by subtracting the mean from each pixel of the input data and then dividing the resultant output by

the standard deviation.

In this experiment, we have collected data from various sources, sizes, and genres in a subtle way. Firstly, for the videos, we have to use the MTCNN network architecture to get the face and the eye points, thereby building the face and eye bounding boxes. Once we have the bounding boxes, we have cropped the eye bounding boxes to shape 56x64 so that the CNN network can, for the most detail, learn the best, which is quite significant for the training (Figure 3.5).

For another dataset which we have collected from the crawl on the internet, Some dataset has size like 185x111 pixels, 104x58 pixels,... That so difficult to take the train, test dataset for the model CNN learning. In this case, we reshape the image to standard shape input to the model convolutional neural network shape 56x64.

The total dataset consists of the training dataset, evaluation dataset, and test

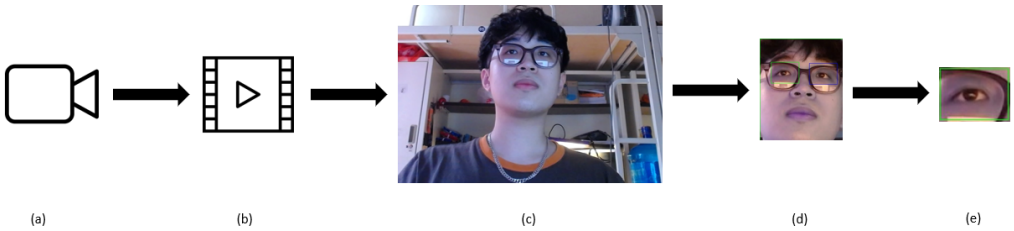

Figure 3.5: (a) Real-time Camera; (b) Video with size 2560x1820 pixels; (c) Split Frame in Video size 2560x1820 pixels; (d) Bounding Box of face and eyes; (e) Crop Bounding box eyes size 56x64 pixels

dataset. The training dataset consists of more than 35000 frames open eyes and more than 35000 frames close eyes. The evaluation dataset consists of 5000 frames open eyes, 5000 frames close eyes, and the test dataset consists of 10000 frames open eyes, 10000 frames close eyes. During training and evaluation, each frame is binary labeled open and closed. The videos are in 2560x1820 pixels, but we are processing crop the eye and reshape it to 56 × 64 pixels, 24 frames per second PNG format without audio to prepare for the next step.

3.3 Convolutional Neural Network for Drowsiness Detection

As I literally mention in section 3.1 we for all intents and purposes have the CNNfor the model eye-tracking driver to mostly detect drowsiness, which kind of is fairly significant. To mostly make model can generally detect pretty high accuracy, the model we actually have some layers specifically likes the 3.6 in a particularly big way.

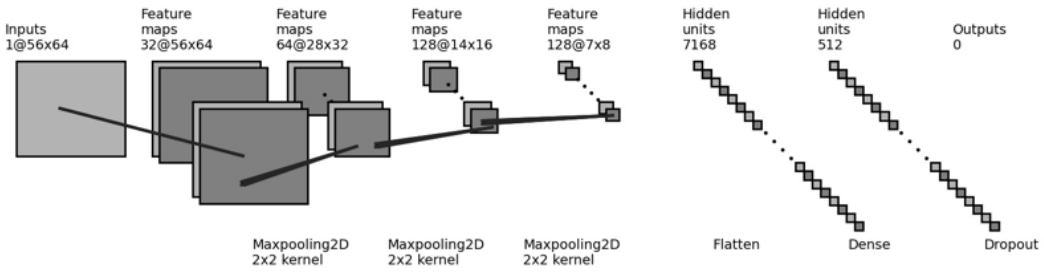

Figure 3.6: Model CNN to training for system tracking eye to detect drowsiness while driving

With the data we have gathered from section 3.2.1 as a figure 3.7 is not eligible to be able to perform the training model step. To be able to make the input for the trainning model, in the first step we converted the image from (56x64x3) with 3 RGB color channels to a Gray image (56x64x1) according to the figure. 3.8.
When the input images have been converted to standard gray as the model, we will transition to the conv2d layer. To prepare the features for image classification well, we will take 32 filters in this first step and a matrix that will scan through the data matrix will have a size of 3x3. To be able to optimize all the features in the image, the distance between 2 kernals when scanning is 1. In this step we use ReLu, between using the sigmod function and tanh we choose relu by because the

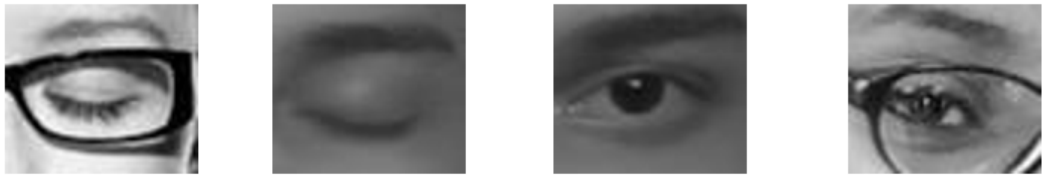

Figure 3.7: Input Dataset Open and Close Eye for the model

Figure 3.8: Data image RGB to Gray

convergence speed is much faster. The ReLU convergence speed is 6 times faster than Tanh (11). This is probably because the ReLU is not saturated at the ends like Sigmoid and Tanh. Input image as figure 3.8 and the out after step through conv2d in figure 3.9.

After we passed conv2d step like figure 3.9 then maxpooling layer with 2x2 kernels will help us get the snow position of features in image space that are no

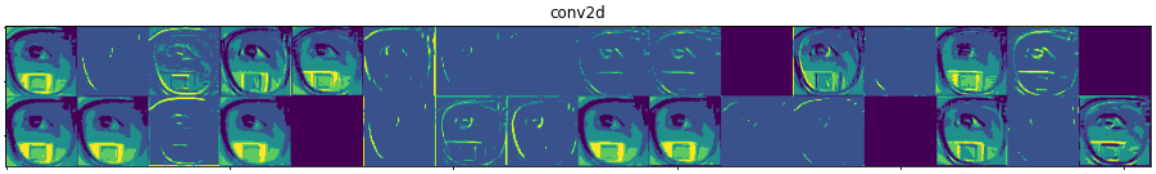

Figure 3.9: First layer: Conv2D

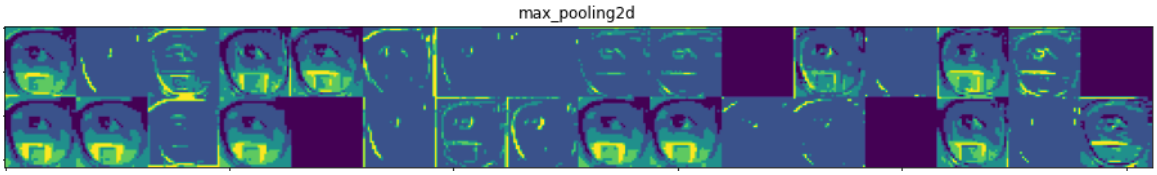

Figure 3.10: Feature after maxpooling

longer needed. instead, the relative position holds sufficient features to classify the object. Besides, max pooling has the ability to reduce the input image dimension from (56x64) to (28x32), limit overfit, and reduce training time. Figure 3.10

The next layers will do the same to prepare for the final layer of the CNN model in the eye-state recognition problem. This stage has the function of converting the symbolic matrix of the preceding layers into a vector containing the probability of the eye state. The CNN model then returns a model that can classify the eye status so that it can provide the system with sleep detection while monitoring the driver's eyes.

3.4 Drowsiness Dectection Tracking Eye System

According to the tech we mentioned above, in this system we will use three main parts. The first technology we use in this system is the MTCNN used for facial recognition, then we will use the trained CNN model to be able to recognize the state of the eye and finally. PERCLOS to be able to identify the driver is in the sleep state or not. Our system will be built in flow as shown in Figure 3.11

3.4.1 Detect Face with Multi-task Cascaded Convolutional Networks

In our system, face detection is located in the head position because if you want to identify the driver's eye, the face detection is very important because if the face cannot be identified, the tracking the driver's eye will not be viable. In this experiment, we used Detect Face with Multi-task Cascaded Convolutional Networks (MTCNN) one of the facial recognition models used by many people today. With MTCNN, we can get the facial landmark and bounding box of the face (figure 3.12). From

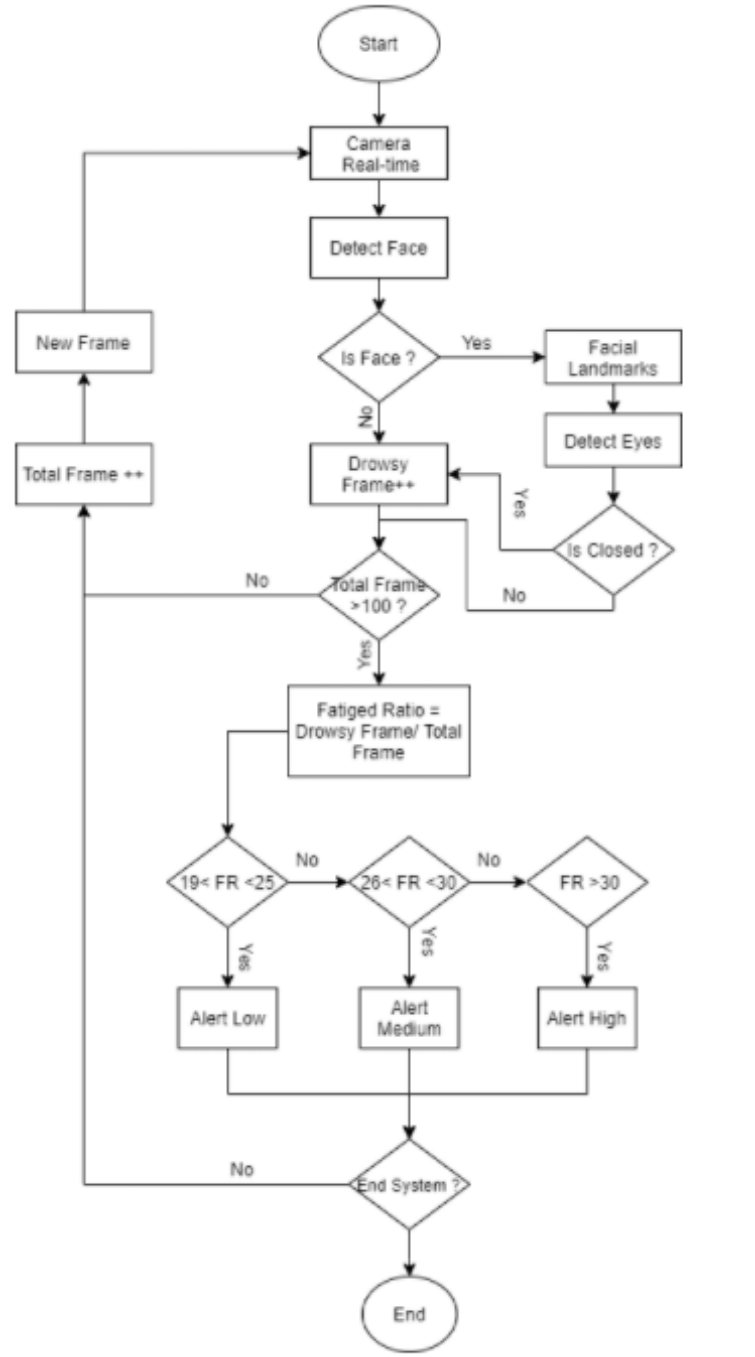

Figure 3.11: Flow of system

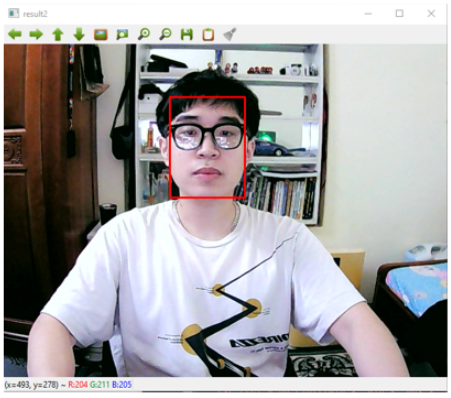

Figure 3.12: MTCNN to extract bounding box of face

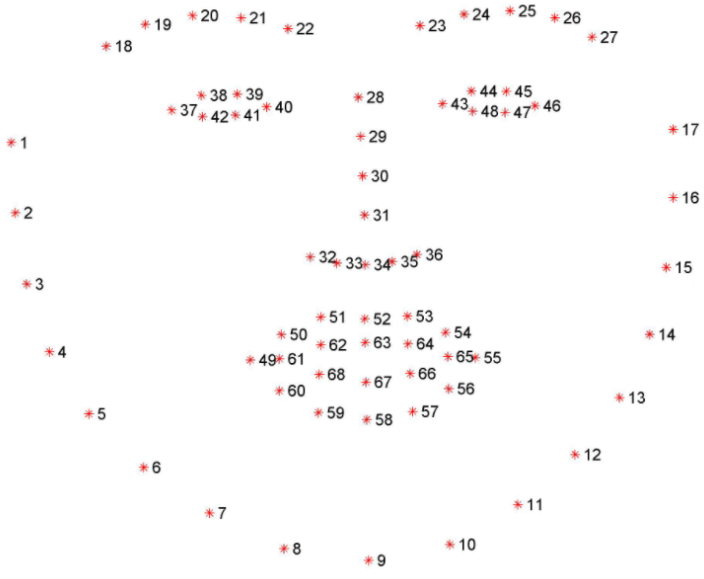

Figure 3.13: Facial Landmarks points

Figure 3.14: Eye when MTCNN and Facial Landmarks extract to predict status

MTCNN we get center point of eye and the facial landmarks (figure 3.13) we get image eye to tracking.

3.4.2 Convolutional Neural Networks Detect Eyes Status

When the Model MTCNN and the facial landmark have obtained the face and image with the shape (56x64) like as figure 3.14, in this step, the model we trained section 3.3 will predict what state the driver's eye is in and from which they are. We can calculate the driver's eye-closing or eye-opening rate to give the most appropriate warning.

3.4.3 PERCLOS and Alert

After the system can detect whether the state of the eye is open or closed, then our system will count the number of frames the camera receives and then apply the PERCLOS algorithm to be able to calculate the ratio. drowsiness while driving. In our system there will be 3 types of warning.

Our first warning is when the driver is in a normal sleep state. With short eye-closing time, continuous frequency, the driver is in a state of lack of alertness. In section 2.5 we have mentioned PERCLOS, in this case when the driver's P ratio is between 19-25 %, we will leave warning low to let the owner know he is sleepy.

Next is that the driver closes his eyes for a very long time and when the mid-level warning driver shows no signs of opening, the driver is in a state of short sleep and in the body can not. resist sleepiness. In this case, the driver's P ratio is between 26-30 %. In this case the driver is having a short nap and the medium alert

with louder volume and greater frequency will be the driver to concentrate again.

The last warning is that when the intermediate warning signals the driver that the driver cannot wake up, our advanced warning can be combined with the slow-down vehicle and can warn for everyone. people around by lights or sound. In this case, the driver's P ratio is over 30 %, which is an alarming case because the driver is too tired and the alarm cannot make the driver awake. In this case, we will warn people around with our headlights or warning lights.

Chapter 4

EXPERIMENTS RESULTS

In this project, we used a computer with an Intel Core i7 - 10700 processor, Camera (Webcam) Thronmax STREAM GO X1 1080P to help stream and record full HD video at 30 frames per second. In this experiment, we want to use pure CPU to embed it on mobile phone or car software at an optimal cost easily.

According to measured data such as the table 4.1, we have tested some cases to optimize real-time image processing to detect the driver is sleepy immediately. We tried with TensorFlow and PyTorch, and the result shows that PyTorch is more efficient.

Even though the CPU used has eight physical cores, it can be immediately noted by the elapsed time, that training with a CPU still takes an extreme amount of time and is not really feasible for fine-tuning in the scope of this project. Even when a CPU consists of multiple cores and is top of the line, high processing times still

Test FPS			
Case	Frame	Time	FPS
Camera normal	100	3.83(s)	26-27
Camera using mtcnn with TensorFlow	100	28.3(s)	3-4
Camera using mtcnn with PyTorch	100	10.6(s)	9-10

Table 4.1: FPS when run with the system testing.

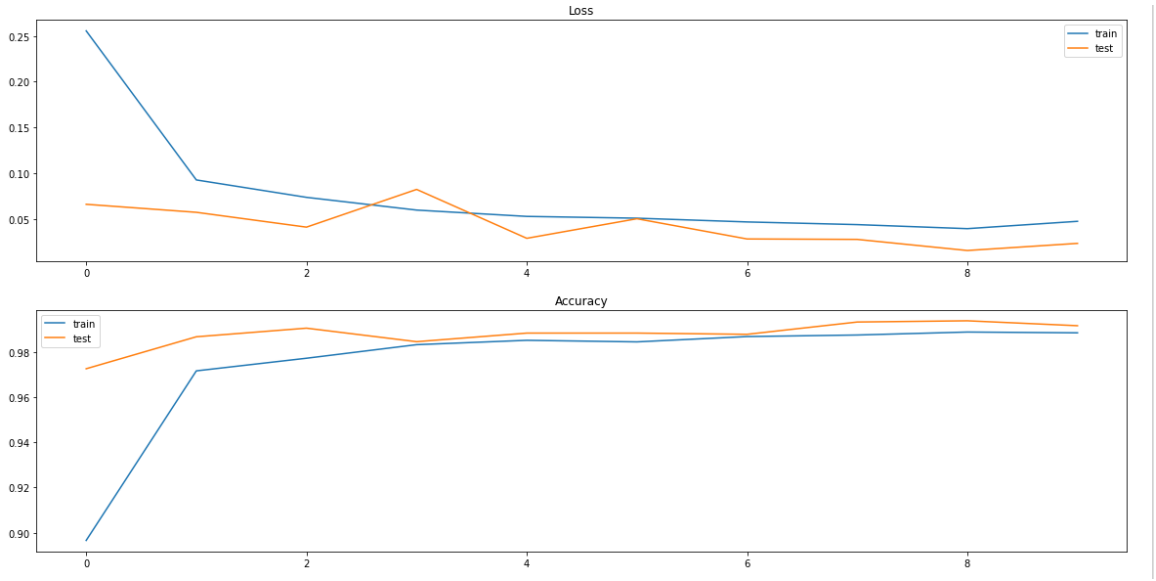

Figure 4.1: Model CNN training with GPU in Google Colab

Test Case		
Case	Amount of People	Accuracy
Bare face	15	92.8%
Wearing Glasses	15	90.4%

Table 4.2: Drowsiness Detection Case Testing.

remain an issue.So we switched to using Google Colab with GPU K80s to optimize the learning model time and possibly help us test faster. With more than 100,000 images from FPT students and a large data source on the internet, the training took about 2 hours. And the results are as shown in figure 4.1

From the results in the figure 4.1 for you, our model is doing very well. We check with 10 epoch, and the number of epochs increases, both Train Loss and Validation Loss decrease, Train Accuracy and Val Accuracy increased. With the model learning so well, our system can detect the status of the driver's eye so that it can alert them if they feel tired and sleepy.

We tested on 15 people at FPT University, and the result can be very good with accuracy greater than 90% according to table 4.2. We also tried on some videos on the Youtube have content fell asleep and the results like figure 4.2

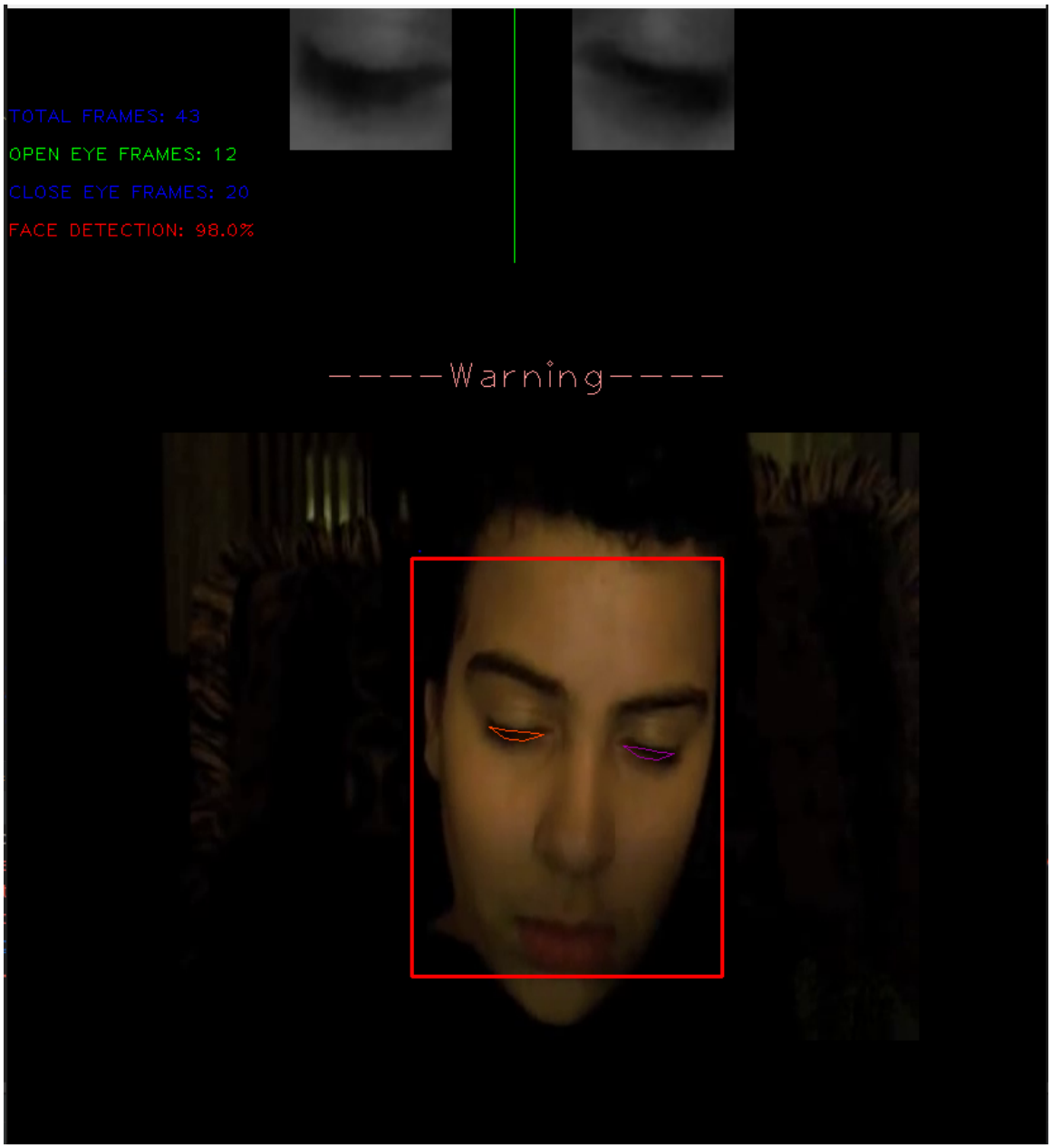

Figure 4.2: Our System tracking eye to detect Drowsiness and Alert

Chapter 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In summary, our driver's drowsiness eye-tracking system is a combination of multiple neural networks. There are three main things that we have presented in this thesis.

Firstly, we presented Neural Network (NN), Convolutional Neural Network (CNN), and Multi-task Cascaded Convolutional Networks (MTCNN). Our thesis clearly and firmly defines its architecture, its learning process, and how to apply it to a supervised learning problem.

Secondly, in this thesis, we have shown flow works of combination deep learning models in each system layer. Each model will have certain strengths and jobs, so we have exploited the model's strengths to make the system work in the best way.

Lastly, our experiments showed that the combination models learn better than the original neural model and achieve higher accuracy. However, it also brings many problems, which we can improve models to solve in the future.

5.2 Future Works

Our system has many limitations that need to be improved to be widely applied in practice. In the future we want to improve the following two things:

The extension ability: Our system is currently running on pure code, so in the future, we want to embed it in some smart devices like phones, car systems so we can monitor the driver's condition. This will help the driver to report his fatigue while driving so that he can find a way to handle it in a timely manner, which can be easily used in traffic to reduce accidents. In some cases, the system may recommend the motel driver or where the driver can stop and take naps to stay awake before returning to the road.

The accuracy ability: We want to find some other models that can increase the exact proportions or incorporate some more facial details, such as the ratio between the driver's eyebrows- eyes, mount- eyes to improve the calculation to helps the driver to be the safest.

References

[1] AGGARWAL, C. *Neural Networks and Deep Learning: A Textbook*, 1st ed. 2018 ed. Springer, 2018.

[2] ALBAWI, S., MOHAMMED, T. A., AND AL-ZAWI, S. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (2017), pp. 1–6.

[3] ARBIB, M. Warren mcculloch's search for the logic of the nervous system. *Perspectives in biology and medicine 43* (02 2000), 193–216.

[4] DANISMAN, T., BILASCO, I. M., DJERABA, C., AND IHADDADENE, N. Drowsy driver detection system using eye blink patterns. In *2010 International Conference on Machine and Web Intelligence* (2010), pp. 230–233.

[5] FUSEK, R. Pupil localization using geodesic distance. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11241 LNCS* (2018), 433–444.

[6] JUNAEDI, S., AND AKBAR, H. Driver Drowsiness Detection Based on Face Feature and PERCLOS. *Journal of Physics: Conference Series 1090* (2018), 012037.

[7] KETKAR, N., AND MOOLAYIL, J. *Convolutional Neural Networks*. 04 2021, pp. 197–242.

[8] KONSTANTINIDIS, D., DIMITROPOULOS, K., AND DARAS, P. A deep learning approach for analyzing video and skeletal features in sign language recognition. In *2018 IEEE International Conference on Imaging Systems and Techniques (IST)* (2018), pp. 1–6.

[9] KRAJEWSKI, J., SOMMER, D., TRUTSCHEL, U., EDWARDS, D., AND GOLZ, M. Steering wheel behavior based estimation of fatigue.

[10] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems 25* (01 2012).

[11] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM 60*, 6 (2017), 84–90.

[12] MARDI, Z., ASHTIANI, S., AND MIKAEILI, M. Eeg-based drowsiness detection for safe driving using chaotic features and statistical tests. *Journal of medical signals and sensors 1* (05 2011), 130–7.

[13] MARQUART, G., CABRALL, C., AND DE WINTER, J. Review of eye-related measures of drivers' mental workload. *Procedia Manufacturing 3* (2015), 2854–2861. 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015.

[14] NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION GOVERNMENT. *Drowsy Driving. https://www.nhtsa.gov/risky-driving/drowsy-driving*.

[15] OLISAH, C., AND SMITH, L. Understanding unconventional preprocessors in deep convolutional neural networks for face identification. *SN Applied Sciences 1*, 11 (2019).

[16] QING, W., BINGXI, S., BIN, X., AND JUNJIE, Z. A PERCLOS-Based Driver Fatigue Recognition Application for Smart Vehicle Space. *2010 Third International Symposium on Information Processing* (2010).

[17] SANIL, N., VENKAT, P. A. N., RAKESH, V., MALLAPUR, R., AND AHMED, M. R. Deep learning techniques for obstacle detection and avoidance in driverless cars. In *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)* (2020), pp. 1–4.

[18] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks 61* (04 2014).

[19] SUN, Y., LIANG, D., WANG, X., AND TANG, X. Deepid3: Face recognition with very deep neural networks.

[20] SUN, Y., WANG, X., AND TANG, X. Hybrid deep learning for face verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38* (10 2016), 1997–2009.

[21] TAIGMAN, Y., YANG, M., RANZATO, M., AND WOLF, L. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708.

[22] WADHAWAN, A., AND KUMAR, P. Deep learning-based sign language recognition system for static signs. *Neural Computing and Applications 32*, 12 (2020), 7957–7968.

[23] WU, Y., AND JI, Q. Constrained joint cascade regression framework for simultaneous facial action unit recognition and facial landmark detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[24] YAMASHITA, R., NISHIO, M., DO, R., AND TOGASHI, K. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging 9*, 4 (2018), 611–629.

[25] ZHANG, K., ZHANG, Z., LI, Z., AND QIAO, Y. Joint face detection and alignment using multitask cascaded convolutional networks. vol. 23, pp. 1499–1503.

[26] ZHANG, S., CHI, C., LEI, Z., AND LI, S. Z. Refineface: Refinement neural network for high performance face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1.