Driver Monitoring with Deep Learning

1st Sheikh Muhammad Adib Bin Sh Abu Bakar
University of Applied Sciences Hamm-Lippstadt
Lippsatdt, Germany
sheikh-muhammad-adib.bin-sh-abu-bakar@stud.hshl.de

***Abstract*—Human error is one of the biggest contributors to car accidents today. For that reason, various kind of system has been built to reduce that error. One of the well-known systems is called the Advance Driver Assistance System (ADAS). Through a safe human-machine interface, ADAS increases car and road safety. Today, with advanced research on computer neural networks, ADAS has been improved with deep learning. This is the main theme of this paper, where a driver monitoring system with deep learning will be discussed. The system is focused on detecting the drowsiness of the driver using deep learning with the proposed method so that appropriate action could be taken to ensure driver safety. The algorithm used for the method will be explained in detail, followed by the experiment ran on the model with the result.**

***Index Terms*—ADAS, drowsiness detection, deep learning**

I. Introduction

Most of the car crash that happened today is due to human error [9] where some of them lead to death. To reduce this occurrence, various methods have been developed including the Advance Driver Assistance System (ADAS), which is developed to reduce the number of car accidents and the seriousness of those that cannot be prevented in order to prevent deaths and injuries. [14], [16].

According to [15] – " 10–15% of car accidents are related to lack of sleep". This is the reason one of the main components of ADAS is detecting driver drowsiness. In recent years, many algorithms have been proposed to detect driver drowsiness.

Drowsiness detection techniques can be classified into three categories: physiological measures, vehicle-based measures, and face analysis [7]. For vehicle-based measures, sensors are installed in the vehicle components to monitor and analyse driving performance and identify driving trends such as steering wheel movement, random braking, and speed changes in order to detect the driver's drowsiness. Adopting vehicle-based solutions have the significant issue that the vehicle's behaviour can change due to poor weather and road conditions [7].

The biggest challenge of using physiological approaches is to ensure driver comfort while wearing sensors, as physiological measurements depend on bodily factors like heart rate, body temperature, pulse rate, and others, as physical conditions change as a driver grows fatigued [7].

Behavioural measures or face analysis detect drowsiness by observing facial expressions and movements using machine learning and computer vision [7]. This technique seems more driver-friendly, Thus becoming the main interest of this paper, where a method for driver drowsiness detection using deep learning is proposed.

This paper is structured as follows: in Section 2, the background on deep learning models and definition of terms used in this paper is presented, followed by related works in section 3. The Details of the proposed methods are presented in Section 4. The experiment on the proposed model is discussed in section 5 with the result. Finally, the conclusion of this paper on Section 6.

II. Background

Before the method used for driver drowsiness detection is explained in detail, some basic terms used in this paper should be first comprehended so that the step used in the method's development can be clearly understood. Two things will be explained, drowsiness and neural network in deep learning, as an overview of the component of the proposed method.

A. Drowsiness

In this paper, a method used for drowsiness detection is proposed. Thus, it is essential to have a clear definition of drowsiness, so that the proper algorithm can be made. Drowsiness in this paper is defined as when a driver closes their eyes longer than usual, which is 2 seconds. It means that detecting the driver's drowsiness is measuring how long he or she closes their eyes. This implementation will be further explained in the proposed method section

B. Deep Learning

Deep Learning or known as deep structured learning [3], which is modelled after the structure and operation of the brain, is a subfield of machine learning approaches that deal with algorithms that use numerous layers to gradually extract higher-level properties from the raw input. Another frequently mentioned advantage of deep learning models, in addition to scalability, is their capacity for automatic feature extraction from unprocessed data, also known as feature learning [10]. A neural network is created to make predictions based on a given input. That is, the neural network is the core of deep learning. Just like humans, before they could make any decision, they should learn how to make the right decision. This phase is called the training phase after the neural network, or the model is completely built. This phase will be detailed in the proposed method section. In this section, the basic building block of a neural network will be explained to give an idea of how a neural network works. The neuron is the basic building block of neural networks in deep learning [6]. The neuron, depict in “Fig. 2”, can have several inputs and outputs just like a human neuron except for this neuron, depict in “Fig. 1”, is a set of a mathematical function [6]. The multiple input can be visualized as shown in “Fig. 3”.

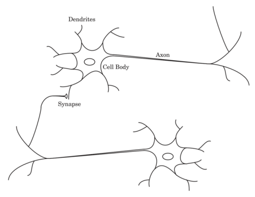

Fig. 1. Human neuron [6]

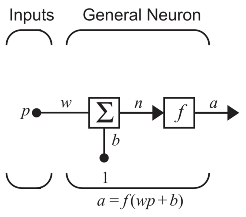

Fig. 2. Neuron with single input [6]

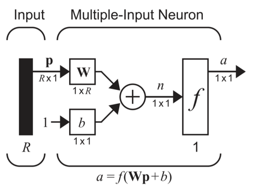

Fig. 3. Neuron with multiple input [6]

Every neuron has its own transfer function or activation function to satisfy some specification of the problem the neuron is attempting to solve [6]. “Fig. 4” shows the example of a transfer function. The linear transfer function is used when the output should be linearly proportional to its input. Neurons with this transfer function are used in the ADALINE networks [6].

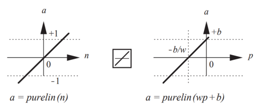

Fig. 4. Linear transfer function [6]

Another transfer function is log sigmoid, visualized in “Fig. 5”, which constricts the output to the range of 0 to 1 from the input that can have any value between plus and minus infinity. Because it is differentiable, the log-sigmoid transfer function is frequently employed in multilayer networks that are trained using the backpropagation process [6].

The hard limit transfer function, visualized in “Fig. 6”, changes the neuron’s output from 1 to 0 depending on whether the function parameter is greater than or equal to 0. This process produces neurons that categorize inputs into two different groups. [6].

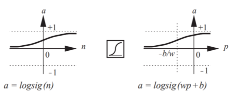

Fig. 5. Sigmoid transfer function [6]

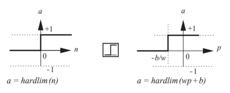

Fig. 6. Hard limit transfer function [6]

As mentioned before, the neuron is the basic building block of a neural network. This building block can be grouped into the form called layer as depicted in “Fig. 7”. Every layer has its own functionalities and name. For instance, a convolutional layer where the neuron acts as a kernel or filter, pooling layer, dropout layer and fully connected layers. These specific layers will be discussed more in the proposed method section.

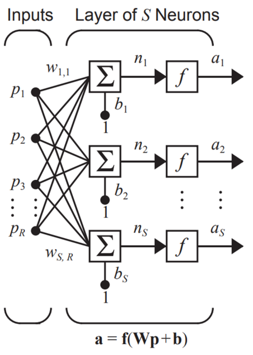

Fig. 7. A layer in neural network [6]

The specific layers are then arranged in a sequential way to form a neural network or network model. For example, a Convolutional neural network (CNN) is widely used in a neural network for image recognition that consists of at least one layer of convolution layer [13].

Aforesaid, a neuron in computer science is nothing more than a set of mathematical functions that have input and output. That means a layer of neuron has a set of input and output as shown in “Fig. 7”. These variables are handled in matrix form where the input matrix will be multiplied by the weight and matrix of the activation function and added with bias before being supplied to the activation function. The output could be scalar or vector, depending on the input matrix and activation matrix. Since the input and output could be vectors that have more than two dimensions, they are called tensors of input or output [13].

III. Related Works

Driver monitoring using deep learning is common these days. Many different methods have been proposed. [12] for instance, use building block as described in “Fig. 8” for their Driver monitoring system. Some blocks or modules depend on the output of others, for example, the input of the blink detection module depends on the output of the face detection module. The implementation of these building blocks makes use of CNNs, which have produced superior outcomes to more traditional image processing or computer vision-based approaches.

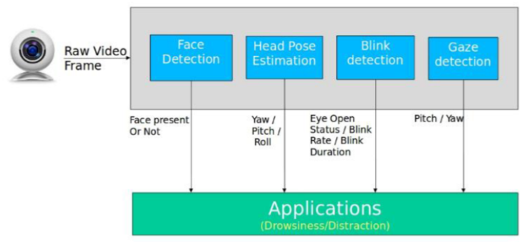

Fig. 8. Architecture used in [12]

A few steps have been taken to optimize the procedure because this model is intended to run on inexpensive embedded devices. The chosen platform is used for the training and inference phases. As a result, the platform can support the produced model’s complexity while maintaining the appropriate level of accuracy.

Pruning and quantization are also used to optimize the selected model. Although post-training methods like weight quantization are likely to produce results that are less accurate than the full-precision version, the accuracy can be increased by retraining the network using the full-precision version’s quantized weights as a starting point.

Utilizing more advanced deep learning techniques, such as replacing the convolution layer with a depth-wise convolution layer, is another way to reduce the model’s complexity. Another method of optimizing for embedded platforms is choosing the appropriate deep learning technology. TensorFlow was chosen for this strategy because it supports remote computation and has effective graph visualization features, which are crucial, especially during training, and It offers C++ and Python APIs and compiles more quickly than most other libraries. Additionally, it includes the (Tflite) inference library, which is lightweight and designed for embedded devices.

This approach also supports multithreading, which helps make the most of multicore processors. Since some of the modules rely on the results of the earlier modules, this must be done strategically. This device is offered for sale and goes by the moniker See ’n Sense for the ARM/iMX platform.

[7] proposes a further deep learning-based solution for driver monitoring systems that focuses on drowsiness detection. This technique utilized a customized YoloV5 pretrained model and Vision Transformers (ViT) to analyse the robust binary image classification model, proposing a behavioural method framework from face detection to drowsiness detection. “Figure 9” depicts the framework. The dataset was increased through image augmentation.

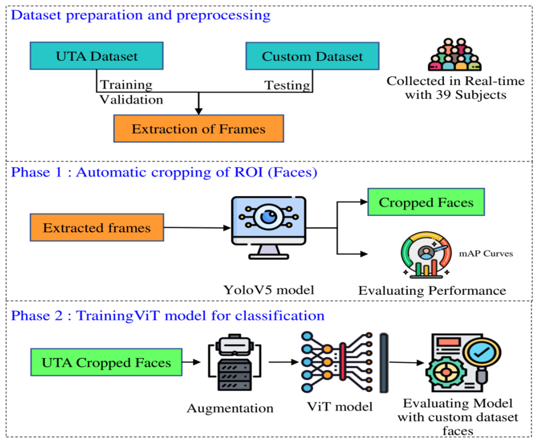
Dataset preparation and preprocessing
UTA Dataset
Custom Dataset
Collected in Real-time with 39 Subjects
Training Validation
Testing
Extraction of Frames
Phase 1 : Automatic cropping of ROI (Faces)
Extracted frames
Cropped Faces
mAP Curves
YoloV5 model
Evaluating Performance
Phase 2 : TrainingViT model for classification
UTA Cropped Faces
Augmentation
ViT model
Evaluating Model with custom dataset faces

Fig. 9. Architecture used in [7]

After numerous thorough analyses, the YoloV5 obtained an accuracy rate of about 90%. The ViT model was evaluated, and the results showed that by using a custom dataset, the framework had achieved good average precision and sensitivity values.

The suggested architecture’s drawback is that a substantial amount of data with labelled scene circumstances is needed in order to train the model. To reduce costs and increase computational efficiency without degrading, the design must be adapted for use in microcomputer systems.

In this paper, a method for drowsiness detection using deep learning is proposed that can improve some aspects of the pervious method, especially in terms of accuracy. For those reasons, the method is also implementing transfer learning to increase the accuracy as implemented in [4] and uses module technique as proposed by [12]. The dataset that will be used will be expanded using the augmentation method as proposed by [7]. The detail about the method will be explained in the proposed method section.

IV. Method

In this section, a proposed method for detecting driver drowsiness using deep learning is discussed. As previously brought up, this method will use module technique, transfer learning and image augmentation. The module technique will allow the proposed neural network to work on fewer cases, which is detecting either the driver’s eyes are closed or open to detect the driver’s drowsiness based on the paper’s drowsiness definition. First, we will use the cascade classifier provided in the OpenCV package to crop the region of interest from the input, which is the image from the camera. The output image, which is the image of the eyes of the driver, will be the input to the proposed model to detect whether the eyes are closed or not. The input image resolution is converted to 80 x 80 with Greyscale and 3 channel to be compatible with the model. If a closed eye is detected for a longer period than usual, an alarm will be executed as a sign that the driver's drowsiness is detected.

The proposed model consists of transfer learning where the inceptionV3 model is used as a based model and combined with a purposed neural network. With the help of transfer learning, knowledge may be adapted or transferred from one set of activities or domains to another. The structure of inceptionv3 is depicted in "Fig. 10" and the part implemented in the proposed method is within the box with a red dotted border, and it consists of 4 main components, which are convolution, maxpool (max pooling), avgpool(average pooling) and concat (Concatenate). These components are an abstraction of layers of neurons or neural networks [13].

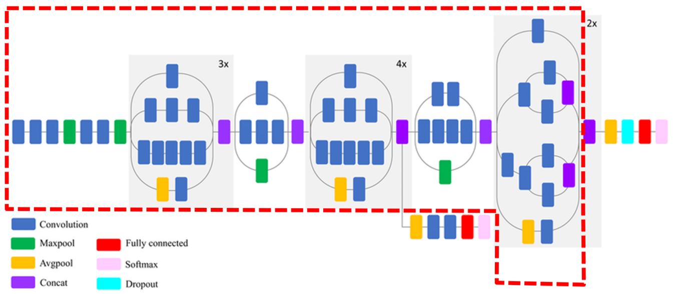

Fig. 10. InceptionV3 structure [8]

The main objective of convolution is to extract features from the input image and produce feature maps. The output feature maps in the initial convolutional layer may learn to detect basic features, such as edges and colour composition variation [11]. By considering the input image as a 3d matrix or tensor where height, width and depth of the image is the parameter, the feature extraction is done by multiplying the matrix with another 3d matrix called filter or kernel as depicted in "Fig 11". The values in the filter tensor are fixed in the way that a certain feature can be extracted, and the filter normally has a smaller size. So, the filter tensor will go through the image tensor by shifting the column and row of the image tensor. The step of shifting is formally called stride. Since the result of the multiplication of a matrix will produce a smaller matrix, the resulting image also has a smaller tensor. However, some convolution layers can maintain the size of the input tensor by expanding the input tensor. This process is known as padding [2].

Pooling is a function where the spatial size of the representation is reduced to reduce the number of parameters and computations in the network. The pooling layer operates on each feature map independently. There are two types of pooling which are max pooling and average pooling [2]. Max pooling is operated by selecting a region with a fixed size in the input tensor and selecting the highest value to form a new tensor as an output. The region then moves by step of a fixed stride. As a result, a smaller tensor is produced. In the average pooling layer, the output is the average value of a selected region.

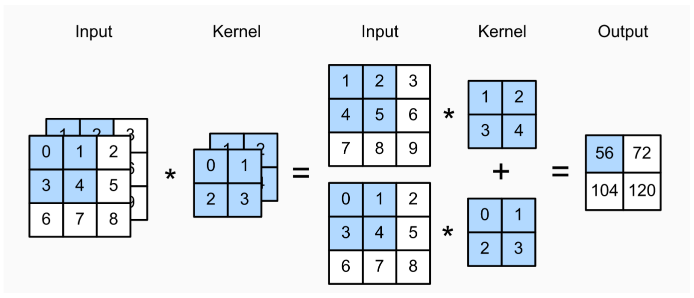

Fig. 11. Example of filter or kernel visualization [2]

A concatenate layer is a layer that combines more than one input tensor to become one output tensor. This output will be used by the next layer that requires a tensor that is bigger than one of the input tensors sizes.

The output from inceptionV3 then becomes the input to the proposed neural network, depicted in "Fig. 12", where the input from the inceptionV3 model is flattened. Since the final output is only a binary decision because only the state of the eyes needs to be detected, whether they are closed or not, the output from the flattened layer is reduced to 64 neurons as an input to the dropout layer to reduce the number of output into two outputs. Aforesaid, every neuron has its own activation function to solve a certain problem. As shown in "Fig. 12", the activation used in the proposed neural network is ReLU and Softmax, where ReLU is used for neurons in the first dense layer and Softmax is used for neurons in the second dense layer.

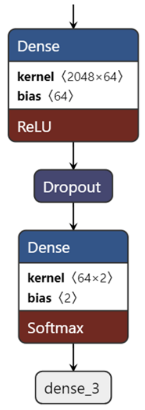

Fig. 12. Proposed Neural Network

V. Experiment

In this section, the proposed method will be evaluated, which is done through the training phase. There are two main processes in the training phase which are training and inference. Aforesaid, the input from every neuron is multiplied by its weight and added with its bias to be an input to the activation function. In the training phase, these parameters are adjusted so that the final output is the same as the expected value. Before further discussion on the training phase, the terms and data used for the training phase should be first comprehended. This data can be image, text, sound, time series and video and called data set [10]. The data set will be split into two sets, one for the training process and another one for the inference process.

The data set used for the proposed model is a set of images from [1] that contain images of closed eyes and, 37751 images of open eyes. A part of them is depicted in "Fig. 13. In total 76408 images, where 80% of it will be used for the training process to find the right value for the weight used in the proposed neural network and 20% for the inference process to evaluate the trained proposed model. Since the proposed model used a transfer learning method, the implemented part from inceprionV3 is not trained.

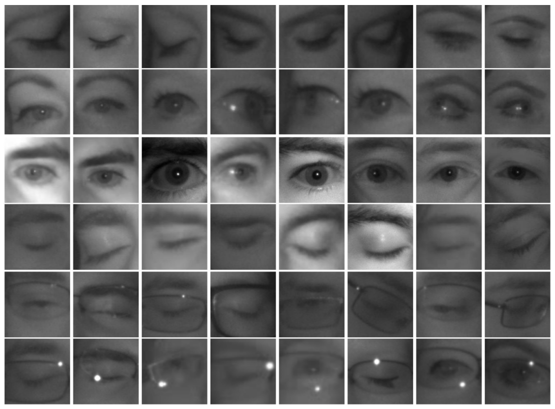

Fig. 13. Example of used dataset from [1]

To handle the huge amount of input, the images are split into batches. The batch size is the number of samples that are passed to the network at once [13]. This is to increase the speed of training because the number of errors that need to be stored is the number of the batch size instead of the error for each image [13].

To increase the accuracy of the model, the data set will pass through the model a few times. This iteration is called epoch [5]. The number of epochs applied for training the proposed model is 5 and the batch size is 8. So, the number of batches for each epoch is 7640.

The proposed method is designed using a python environment and the API used for the model is from TensorFlow[1]. The training phase was executed on a computer with the 64-bit operating system, x64-based processor, Intel® Core™ i7-1065G7, 1.50GHz and 16 GB RAM. Training a network means nothing more than solving a complex optimization problem [10]. At first, the value of weight and bias for every neuron is randomly assigned. After that, an image is given to the model as input. The final output value is compared with an expected value, where the difference value or known as the loss value is recorded. The loss values are used by the loss function to improve the value of the weight in a way that the loss value can be reduced in the future. The lost function that is used in the proposed method is categorical cross-entropy. In the training process, it could happen that the model is "overfitting". This means that the model has high accuracy on trained data but low accuracy in a real environment. To reduce the overfitting, the learning rate is reduced. The algorithm used to achieve that is called an optimizer. For the proposed model, the optimizer used is Adam. The detail about Adam optimization can be found at [11].

[1]TensorFlow is an API library for deep learning software that can be found at https://www.tensorflow.org/

The trained model then proceeded to the inference process where the model accuracy is evaluated, and the remaining data set is used. The result from the whole training phase is shown in "Fig. 14", where the model has an accuracy of 95% on the final epoch. The blue line is the value during training and the orange line is the value during validation or inference. The loss on each epoch is visualized in "Fig. 15". "Fig. 16" and "Fig. 17" visualize the accuracy and loss for every iteration.

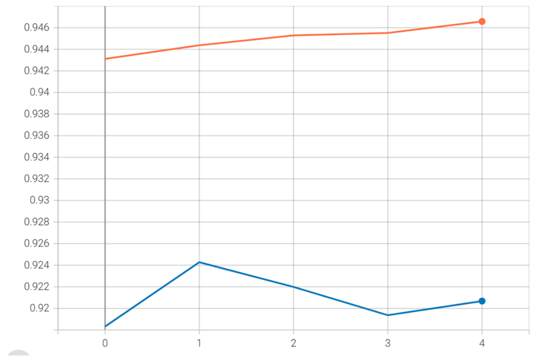

Fig. 14. Model accuracy for every epoch

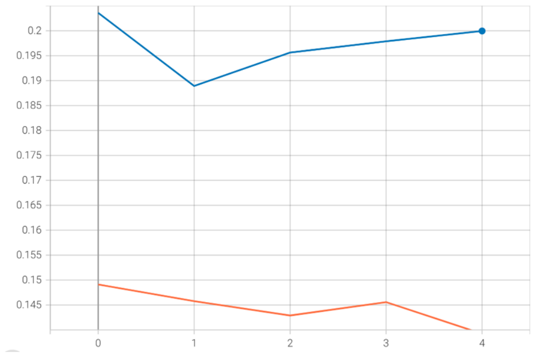

Fig. 15. Model loss for every epoch

VI. Conclusion

In this paper, a method to detect driver drowsiness is proposed as a part of the Advance Driver Monitoring System. With an adaptation of machine learning and working with blocks in a hierarchy, this method has reached its best accuracy, which is 95%. The component or types of layers used in the inceptionV3 and proposed neural network are also introduced in this paper.

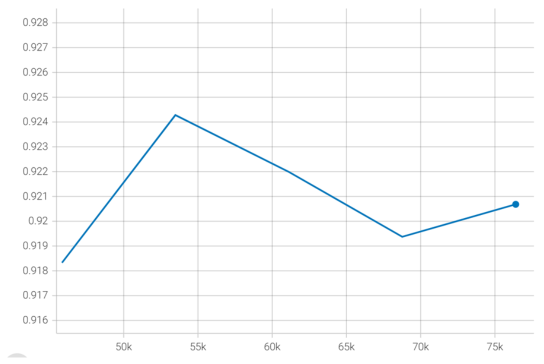

Fig. 16. Model accuracy for every iteration

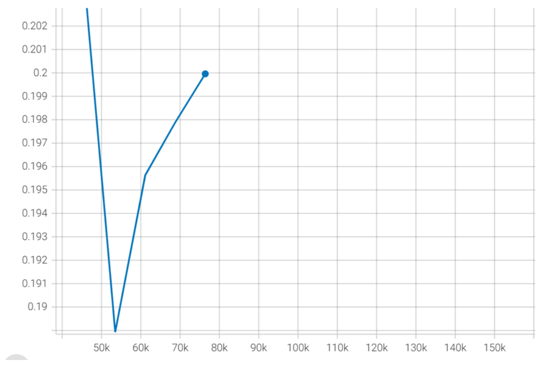

Fig. 17. Model loss for every iteration

The external and internal behaviour and the characteristic of each layer have been discussed to show how the proposed method works. A proper method used for training the proposed model is also analysed, where the internal result of the training process and inference process are explained and visualized.

Although the proposed strategy is not cutting-edge, it has produced some great outcomes. Despite the model's 95% accuracy, several enhancements could be done, such as reducing the complexity of the model, especially on the chosen model (in this case inceptionV3) so that the model can turn on a low-cost system without jeopardizing the accuracy.

Therefore, putting these concepts into practice will be the first step. Following the successful completion of these processes, additional improvements can be made while the system is being tested in a real-world environment.

Acknowledgment

This work was partially supported by the University of Applied Science Hamm-Lippstadt.

References

[1] MRL eye dataset | MRL.

[2] Mu Li Alex J. Smola Brent Werness Rachel Hu Shuai Zhang Yi Tay Anirudh Dagar Yuan Tang Aston Zhang, Zack C. Lipton. *Dive into deep learning*. 2021.

[3] Li Deng. An overview of deep-structured learning for information processing. In *Proc. Asian-Pacific Signal & Information Proc. Annual Summit & Conference (APSIPA-ASC)*, pages 1–14, 2011-10. Edition: Proc. Asian-Pacific Signal & Information Proc. Annual Summit & Conference (APSIPA-ASC).

[4] Zhou Dongmei, Wang Ke, Guo Hongbo, Wang Peng, Wang Chao, and Peng Shaofeng. Classification and identification of citrus pests based on InceptionV3 convolutional neural network and migration learning. In *2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*, pages 1–7, 2020-11.

[5] Frank Gaillard. Epoch (machine learning) | radiology reference article | radiopaedia.org.

[6] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, and Orlando De Jésus. *Neural network design*. Martin T. Hagan, 2nd edition edition, 2014.

[7] Ghanta Sai Krishna, Kundrapu Supriya, Jai Vardhan, and Mallikharjuna Rao K. Vision transformers and YoloV5 based driver drowsiness detection framework. 2022. Publisher: arXiv Version Number: 1.

[8] Masoud Mahdianpari, Bahram Salehi, Mohammad Rezaee, Fariba Mohammadimanesh, and Yun Zhang. Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. 10(7):1119, 2018-07-14.

[9] David Matine. What percentage of car accidents are caused by human error? | pittsburgh law blog, 2021-09-22.

[10] Umberto Michelucci. *Advanced applied deep learning: Convolutional Neural Networks and object detection*. Apress, 2019.

[11] Santanu Pattanayak. *Intelligent projects using python: 9 real-world AI projects Leveraging Machine Learning and deep learning with TensorFlow and keras*. Packt Publishing, 2019.

[12] Nirmal Kumar Sancheti, Manjari Srikant, and Krupa H Gopal. Camera based driver monitoring system using deep learning. page 5.

[13] Mohit Sewak, Md Rezaul Karim, and Pradeep Pujari. *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python*. Packt Publishing, 2018-02-27.

[14] Aleksandra Simic, Ognjen Kocic, Milan Z. Bjelica, and Milena Milosevic. Driver monitoring algorithm for advanced driver assistance systems. In *2016 24th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE.

[15] Marjorie Vennelle, Heather M. Engleman, and Neil J. Douglas. Sleepiness and sleep-related accidents in commercial bus drivers - sleep and breathing, Jul 2009.

[16] Lishengsa Yue, Mohamed A. Abdel-Aty, Yina Wu, and Ahmed Farid. The practical effectiveness of advanced driver assistance systems at different roadway facilities: System limitation, adoption, and usage. 21(9):3859–3870.