Data Augmentation for Deep Learning Algorithms that Perform Driver Drowsiness Detection

Ghulam Masudh Mohamed, Sulaiman Saleem Patel* and Nalindren Naicker
Department of Information Systems, Durban University of Technology
Durban, South Africa

***Abstract*— Driver drowsiness is one of the main causes of driver-related motor vehicle collisions, as this impairs a person's concentration whilst driving. With the enhancements of computer vision and deep learning (DL), driver drowsiness detection systems have been developed previously, in an attempt to improve road safety. These systems experienced performance degradation under real-world testing due to factors such as driver movement and poor lighting. This study proposed to improve the training of DL models for driver drowsiness detection by applying data augmentation (DA) techniques that model these real-world scenarios. This paper studies six DL models for driver drowsiness detection: four configurations of a Convolutional Neural Network (CNN), two custom configurations as well as the architectures designed by the Visual Geometry Group (VGG) (i.e. VGG16 and VGG19); a Generative Adversarial Network (GAN) and a Multi-Layer Perceptron (MLP). These DL models were trained using two datasets of eye images, where the state of eye (open or closed) is used in determining driver drowsiness. The performance of the DL models was measured with respect to accuracy, F1-Score, precision, negative class precision, recall and specificity. When comparing the performance of DL models trained on datasets with and without DA in aggregation, it was found that all metrics were improved. After removing outliers from the results, it was found that the average improvement in both accuracy and F1 score due to DA was +4.3%. Furthermore, it is shown that the extent to which the DA techniques improve DL model performance is correlated with the inherent model performance. For DL models with accuracy and F1-Score ≤ 90%, results show that the DA techniques studied should improve performance by at least +5%.**

Keywords*—Data augmentation; deep learning; computer vision; drowsiness detection; road safety

I. INTRODUCTION

Road accidents represent a major socio-economic challenge for individuals, industries, and nations [1]. Commuters involved in road accidents are affected in a variety of ways; such as death, sustaining physical injuries, psychological trauma, as well as incurring financial burdens from damage to property [1-4]. For industries, road accidents adversely affect supply chain performance and logistics, reducing operational efficiency [5-7]. The net result of this adversely impacts the economy of a country. Furthermore, for national authorities, road accidents cause traffic congestion; resulting damage to infrastructure and increased environmental pollution. Road accidents are a greater concern in developing countries, wherein more than 90% of accidents result in fatalities [1]. Of all developing countries, the World Health Organisation reports that South Africa has the poorest road safety record, with approximately 14 000 deaths per annum and an accident fatality rate of 3.2% [2, 8, 9].

The factors that cause road accidents need to be identified before an effective solution can be developed. Studies, such as those presented by Machetele and Yessoufou [1] and Verster and Fourie [2], highlight that driver-related accidents account for 80% to 90% of fatal road accidents. A key cause of driver-related accidents is drowsiness (which may result from excessive alcohol consumption), as this impairs a person's concentration and focus [2, 10]. The detection of driver fatigue or drowsiness is hence essential towards improving road safety and reducing the accident rate [11, 12].

In light of the fourth industrial revolution, technology is becoming more ubiquitous and there is growing motivation to utilize artificial intelligence and machine learning to solve social problems, such as driver drowsiness detection. To this end, there have been a range of studies that apply deep learning (DL) techniques to solve the problem of driver drowsiness detection [13-19]. DL is a subset of machine learning that mimics the neural network of the human brain, thus creating an artificial neural network [14]. Artificial neural networks comprise of multiple nodes that model neurons of the human brain, which are organized into layers [20]. Data is propagated from the input layer to the output layer. These artificial neural networks have the potential to solve regression and classification problems, including image classification problems [20, 21]. In the context of image classification, each layer trains upon the output of the previous layer, enabling latter layers to identify more intricate elements of the images [21].

At a technical level, the aforementioned studies perform driver drowsiness detection by considering images of a driver's eye, and using DL algorithms to determine the eye state (i.e. whether the eye is opened or closed). By applying this technology to frames from a video feed of the driver, it is possible to determine whether eyes are closed for extended periods of time, which is an indicator of drowsiness. Some of the DL algorithms used in literature include: (i) convolutional neural networks (CNNs) of different configurations [14-16, 18, 22, 23]; (ii) the multi-layer perceptron (MLP) [13, 24]; (iii) the respective Visual Geometry Group 16 (VGG16) [25, 26] and 19 (VGG19) [17, 26] models; as well as (iv) the generative adversarial network (GAN)[27]. The reported accuracies of the models in these studies range between 75% and 96%.

Despite the high accuracies reported in the studies, real-world challenges during implementation were reported that adversely affected the accuracy of the trained models. Among these challenges were: (i) poor lighting, where lighting is either too bright or too dim [13, 14, 17, 19]; (ii) changes to the driver's seat position [22]; (iii) a change in the angle of the driver's face while driving [13, 22] the use of spectacles and/or sunglasses by drivers [14, 17-19, 24].

In this paper, the authors proposed to address these real-world challenges by performing data augmentation (DA) on the training image sets that are input into DL models for driver drowsiness detection. DA techniques introduce artificial images that simulate real-world effects [28], such as different lighting environments and changes to face orientation. This study also uses a training dataset containing images of drivers with and without eyewear to address the challenges associated with drivers wearing spectacles or sunglasses. The DA techniques are tested on CNN models, GAN models, MLP models and both the VGG16 and VGG19 models. Hyperparameter tuning is performed on all models to optimize their learning rate and enhance their overall performance. Literature has shown that careful selection of hyperparameters has a significant impact on model performance [28, 29]. The effect of the DA is evaluated by comparing the performance of models trained with and without DA in with respect to the following metrics: (i) accuracy, (ii) precision, (iii) negative class precision, (iv) recall, (v) specificity, and (vi) F1-score. It is hypothesized that the use of DA will result in improved performance of all models.

It is noted that previous studies in literature [14, 25, 27] have incorporated the use of DA in improving the performance of their specific driver drowsiness detection models. However, to the best of the authors' knowledge, there are no comprehensive studies that investigate DA techniques for a wide range of DL algorithms in the context of driver drowsiness detection, as is done in this paper.

The research in this paper makes the following contributions:

1) Presenting an overview of DA techniques to model the specific real-world scenarios that cause challenges for driver drowsiness detection systems.

2) Studying the DA techniques on a wide range of DL models that perform driver drowsiness detection and statistically analyzing the effects of the DA techniques.

3) Demonstrating the extent to which the DA techniques studied are able to improve DL models that perform driver drowsiness detection and proposing a design guideline for DL model developers on that conditions under which the DA techniques should be considered.

The rest of this paper is organized as follows. In Section II, a review of existing literature was presented. Section III presents the materials and methods used in this study, including providing an overview of a real-world drowsiness detection system. In Section IV the results of the investigations are presented and finally, conclusions and insights that were drawn from this study are presented in Section V. Section V also makes recommendations for future work.

II. Related Work

This section reviews the DL algorithms that have been extensively used in previous studies, to implement models and applications, for drowsiness detection in motorists.

A study by Jabbar et al. [14] proposed a drowsiness detection system that could be implemented on the driver's dashboard, using an Android phone. The system was able to predict the drowsiness of the driver based on their eye state. This study made use of a CNN network to implement a binary classification model that was able to classify the drowsiness in facial images. Data augmentation techniques were applied to the images, before they were trained on the model. The Dlib C++ library was used to extract the driver's facial landmarks from the images. These facial features were fed into the algorithm for training. The dataset was created using the extracted eye features. This model achieved an accuracy of 83.3%. A similar study by Zhang, Su, Geng and Xiao [18] was conducted to detect the drowsiness of a person, using the eye state. This proposed model was implemented on an Infrared video camera. The AdaBoost algorithm was used to extract facial landmarks from the images. The extracted eye landmarks were used to create the image dataset, to train the model on. The CNN model was used as the binary classifier for drowsiness. An accuracy of 95.8% was achieved by this study.

Sharan, Viji, Pradeep and Sajith [15] proposed a similar drowsiness detection system to Jabbar et al. [14] that could be implemented on the driver dashboard. However, this study proposed that a Raspberry Pi camera module be used to capture the drivers face. The drowsiness prediction was also based on the eye state. The Haar Cascade classifier was used for facial extraction during the implementation of this system and the CNN network was implemented as the binary classifier. Contrast Level Adaptive Histogram Equalization was applied to remove the noise and improve the picture quality, before they were trained on the CNN model. The CNN model was trained on an existing dataset, comprising of eye images. The study by Seetharman, Sridhar and Mootha [22] made use of a CNN network to classify the drowsiness in images. The prediction was based on the eye and mouth state of the extracted faces. The Dlib library was utilized to extract the facial regions from the images, similar to the study done by Jabbar et al. [14]. A dataset for the model was then generated using the extracted eye regions. The trained CNN model achieved an accuracy of 92.4%. In addition, this proposed model was intended to be implemented on a dashboard video camera. Chirra, Uyyala and Kolli [16] proposed a similar model for drowsiness detection, as a CNN network was used to predict the drowsiness in images. The eye state was the metric for prediction, with the Viola-Jones algorithm used to extract the facial landmarks from the images, during the implementation of this system. An existing dataset of eye images was used to train the CNN model. The model produced an accuracy of 96.42%. This model was also proposed to be implemented on a video camera for drowsiness detection, like the study conducted by Seetharman, Sridhar and Mootha [22].

A model using the VGG 19 model to detect driver drowsiness, based on the eye state, was proposed by Hashemi, Mirrashid and Shirazi [17]. This study made use of the Viola-

Jones algorithm to extract the facial landmarks from the images. The extracted eye landmarks were then used to create the dataset for this model. The Viola-Jones algorithm has been utilised in previous work [16]. This model obtained an accuracy of 94.96%, with its intended application in driver dashboard monitoring. A study by Ahuja, Saurav, Srivastava and Shekhar [26], proposed an approach to improved drowsiness detection, by using a knowledge distillation technique to reduce the size of DL models, whilst maintaining high accuracy. A large model will have high memory consumption and longer response times. Therefore, there was a need to reduce the size of the DL model. The Histogram of Gradient algorithm was used to extract the facial regions from the images, during system implementation. VGG19 and Visual Geometry Group 16 (VGG16) were the algorithms used to train their respective models, to classify the drowsiness in images. These models were trained on an existing dataset, consisting of eye images. The predictions were based on the eye state for both models. The VGG19 and VGG16 models, obtained the accuracy of 92.5% and 95% respectively.

Bajaj, Ray, Shedge, Jaikar and More [25] proposed a real-time drowsiness prediction system that will be implemented on an Android application, to monitor the driver's face from the dashboard. This system can predict the drowsiness using the driver's eye state. A comparative analysis of three DL algorithms, specifically: Inception, ResNet-50 and VGG 16 were performed. Data augmentation techniques were applied to the images, before they were trained on the models. The models were trained on an existing dataset, comprising of face images. The accuracy achieved by the Inception, ResNet-50 and VGG 16 models were 89%, 56% and 91%, respectively.

A study by Jabbar et al. [13] proposed a system for drowsiness detection that could be implemented on an android application, for dashboard monitoring. The prediction of this system was based on the driver's eye state. The Dlib C++ library was used to extract the person's facial landmarks from the images. This library has been used for facial feature extraction in previous work [14,25]. These facial features were used to create the dataset, which was fed into the MLP algorithm for training. The model was able to classify a driver as either drowsy or non-drowsy. An accuracy rate of 80.92% was achieved by this model. A similar study by Ghourabi, Ghazouani and Barhoumi [24] made use of the MLP algorithm to detect drowsiness in the images. The eye and mouth state were used to classify the drowsiness. The Histogram of Gradient algorithm was used to extract the facial regions from the images. These extracted facial regions were used to create the dataset that was fed into the model for training. The model is intended to be implemented for dashboard monitoring. This study obtained an accuracy rate of 74.9%.

Ngxande, Tapamo and Burke [27] proposed a framework to reduce the biasness of a model during the training process. A Generative Adversarial Network (GAN) model was trained on an image dataset. This model made predictions using facial landmarks and the eye state in particular. The extracted facial landmarks were used to create the dataset for model training. Data augmentation techniques were applied to the images before they were loaded into the GAN model. This helped to improve the performance of the binary classification model. An accuracy rate of 91.62% was achieved by this model.

Many of the studies have used facial and eye extraction algorithms, to create image datasets from real-time data, to train their models on. However, this study aimed to use existing datasets that were available online, to train the DL models. The reason for this was because, this study aimed on improving the performance of trained models, regardless of the source of data. Therefore, no facial and eye extraction algorithms were used on real-time data, in this study.

Literature has shown that many drowsiness detection models faced issues with prediction accuracy, due to poor lighting and the use of sunglasses [13,14,17,18,24]. The other challenge that affected accuracy was the positioning of the driver's face [13,22]. Another gap identified is the lack of pre-processing and data augmentation applied on the data before training. Data augmentation was used in [14,25,27], to create more comprehensive models that exhibits improved performance. DA was used to remove biasness from the models, thus improving the performance. However, not many of the previous studies have comprehensively studied DA to model real-world scenarios to improve model performance, on a wide range of DL algorithms that detect driver drowsiness, as done in this study.

Therefore, this study aimed to develop an improved approach towards drowsiness detection by using data augmentation. Data augmentation techniques were used to create training data that replicate real-life scenarios that correlate with the challenges faced in previous studies.

III. Materials and Methods

This section first provides an overview of a real-world driver drowsiness detection system and isolates the role of the DL algorithms that this study focuses on. The data sources and DA techniques utilized in this paper are then discussed. Thereafter, a technical summary of the DL algorithms considered is provided, along with the parameters used in this study. Finally, the authors present the different evaluation metrics that are used to quantify the performance of the DL algorithms.

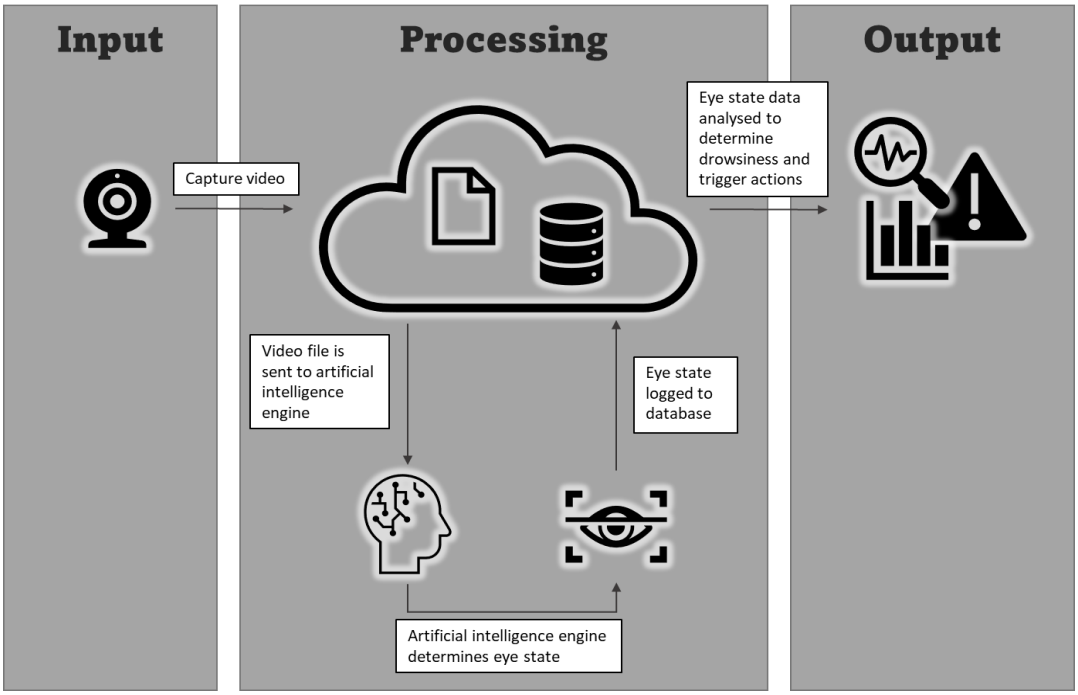

Fig. 1. Overview of a real-world driver drowsiness detection system

A. An Overview of a Real-World Drowsiness Detection System

Fig. 1 illustrates the process flow for a real-world driver drowsiness detection system. The process starts with a camera that captures a video of the driver's face, which serves as the input to the system. The camera can either be mounted to the dashboard or steering wheel of the vehicle. The captured video is then stored on cloud-hosted infrastructure, typically in some form of unstructured blob storage.

At the start of the processing stage of the system, the video file is passed on to an artificial intelligence engine, consisting of three sub-units. The first sub-unit extracts individual frames from the video file, which will then be treated as a series of sequential images. The second sub-unit uses image detection techniques to isolate the eye from each image of the driver's face. This produces a series of sequential images of the driver's eyes. Finally, the third sub-unit utilizes a pre-trained DL model to analyze the images and determine the state of the driver's eye (open or closed) in each frame. The eye state determined in each frame is then logged in a database, which is also typically cloud-hosted.

In the final stage of the system, the eye states stored in the database are analyzed and interpreted to detect the drowsiness of the driver. Drowsiness detected when the driver's eyes are in the 'closed' state for extended periods (multiple consecutive frames from the video feed).

B. Design and Configuration of Study

The research presented in this study focuses on the third sub-unit of the artificial intelligence engine, viz. the DL algorithm that determines the driver's eye state, as described in Section III.A. Hence, for the experiments conducted, the inputs in this study were images of a driver's eye and the outputs were a categorical variable indicating the eye state. A binary categorical output was used, with the positive class label indicating the "open" eye state and the negative class label indicating the "closed" eye state. The experimental configuration used is depicted in Fig. 2.

In performing the experiments, appropriate datasets of eye images were first sourced. In selecting the datasets, the authors ensured that images where the eye was partially obscured by eyewear (spectacles or sunglasses) were included. By doing this, the DL models would learn to distinguish between eye states irrespective of the use of eyewear.

The datasets were then split into training and testing data using an 80:20 ratio. A copy of the training dataset was created, and data augmentation techniques were performed to model the real-world challenges of eye orientation and lighting conditions. Two DL models were trained: one was trained on the original (pre-treatment) training dataset, and the other was trained on the modified (post-treatment) training dataset. Depending on the architecture of the DL algorithm being investigated, any necessary data-shaping modifications were made to the images from the dataset.

The pre-treatment and post-treatment DL models were applied to the testing dataset to evaluate and compare their performance. As was the case with the training datasets, any modifications to the testing dataset required by the DL model architecture were made.

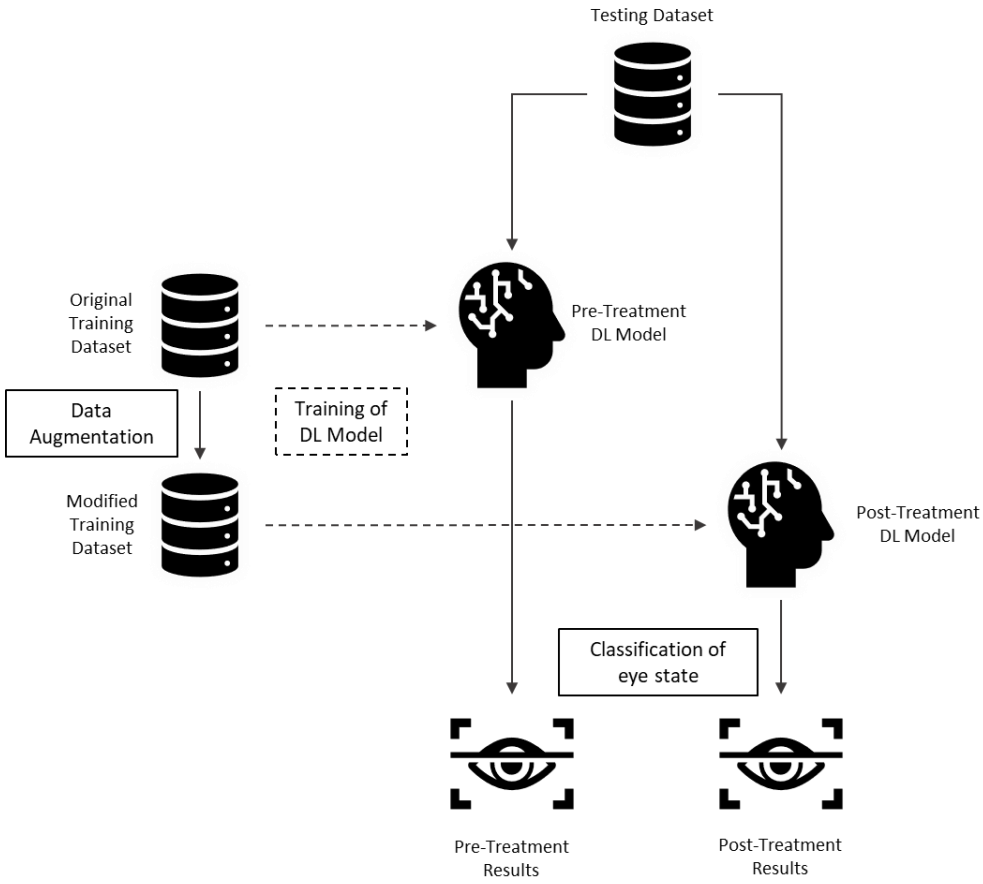

Fig. 2. Configuration of study

TABLE I. PROPERTIES OF DATASETS

Property	Dataset 1	Dataset 2
Total number of images	2 500	1452
Eye Opened	1 250	726
Eye Closed	1 250	726
Image size (pixels)	96 × 96	256 × 256
Colour/Greyscale	Greyscale	Colour
File format	Portable Network Graphics (PNG)	Joint Photographic Experts Group (JPEG)
File compression	Lossless compression	Lossy compression

The experiments were done using pre-built Python libraries on the Jupyter Notebooks development environment. A personal computer equipped with 8 gigabytes of random-access memory, an Intel Core i5-7200U processor and a 64-bit Windows 10 operating system.

1) Selection of datasets: There were two datasets utilised in this study, which were obtained from online repositories [30, 31]. Both datasets contained images of human eyes with and without eyewear, and images labelled according to the eye state. The properties of the datasets are presented in Table I.

The balanced distribution of eye states was preserved when splitting each of the datasets into respective training and testing datasets, using an 80:20 ratio. The Scikit-learn Python library was used to implement the data splitting.

When exploring the datasets, it was also noted that both sets of data contained images from a diverse range of ethnicities. Different skin tones and complexions were noted, as well as different eye shapes. The authors further observed that among female eyes, the extent to which make-up such as eyeliner and false eyelashes were used differed.

2) Data augmentation and pre-processing: Data augmentation improves model performance by generating variations of training data [14]. This reduces overfitting and improves the model's ability to make generalizations [14, 32]. The specific augmentations performed in this study were designed to simulate real-world scenarios and overcome some of the challenges indicated in literature.

The ImageDataGenerator class within the Keras library for Python [33] was used to implement pre-processing and DA in this study. The ImageDataGenerator class supports DA in real-time and makes sure that the model is trained with different variations of images during each training iteration (epoch) [34, 35].

The following pre-processing and data augmentation techniques were applied:

a) Brightness adjustment: Multiple studies in literature have shown that poor lighting conditions had a negative impact on the accuracy of DL models for driver drowsiness detection [13, 14, 17, 19]. While driving, ambient lighting conditions can change due to environmental conditions such as the time of day and the weather. For example, driving at night results in a very low brightness conditions and driving in bright sunshine results in very high brightness conditions. While driving, it is also possible for lighting conditions to change rapidly, such a when driving under a bridge/overpass on a sunny day or through the shadow cast by a building or other large structure.

To model scenarios with different lighting environments, this study applied a randomized change to image brightness when augmenting images. This is implemented through adding a constant, δ, to all pixels in the image. The brightness adjustment function is mathematically described as:

$$f_{\mathrm{b}}(p) = \max(0, min(255, p + \delta)) \quad (1)$$

In (1), p is the value of the pixel and falls in the range $0 \leq p \leq 255$. Positive values of δ increase the brightness of the image while negative values of δ decrease the brightness of the image. The $\max(\cdot)$ and $\min(\cdot)$ functions ensure that the brightness-adjusted pixel value remains in a valid range.

b) Horizontal flips: The shape of a human eye may differ slightly between the left eye and the right eye. Creating artificial data by flipping the horizontal orientation allows the DL model to be trained to analyze either eye of the driver.

c) Rotation, translation and zoom: Literature showed that changes to the driver's face orientation was a real-world scenario that adversely affected the performance of DL models [13, 21]. Therefore, in this study, rotation, translation shifts and zoom transformations were used to model changes to the driver's face orientation. Rotation and translational shifts are useful to simulate movement of a driver's head while travelling. Zoom transformations model a change in depth between the camera and the driver's face, which may result from the driver changing their seat position or posture.

d) Normalization, centering and standardization: Normalization and standardization improve the learning rate and reduces the number of epochs required to train a DL model [36, 37]. These processes ensure that no individual input pixel dominates performance [38]. This is done by mathematically adjusting data such that it follows a Gaussian distribution with zero mean and unit variance [39].

Normalization involves rescaling the value of pixels to have a unit maximum, which reduces the computational power required to train the DL model. As all pixels have the same maximum value ($p = 255$), the normalization function is described by [36]:

$$f_{\mathrm{n}}(p) = \frac{p}{255} \quad (2)$$

Centering ensures that the data has a mean of zero, while standardization ensures that the data has a unit variance [36]. Setting these statistical properties of the data improves the rate at which a DL algorithm converges when training, as well as increasing model accuracy by eliminating statistical bias.

Centering and standardization can be applied to data in with respect to individual images (sample-wise) or with respect to the entire set of images (feature-wise). The functions for sample-wise centering (sc), feature-wise centering (fc), sample-wise standardization (ss) and feature-wise standardization (fs) are [39]:

$$f_{\mathrm{sc}}(p) = p - \bar{p}_{\mathrm{I}} \quad (3)$$

$$f_{\mathrm{fc}}(p) = p - \bar{p}_{\mathrm{D}} \quad (4)$$

$$f_{\mathrm{ss}}(p) = \frac{p}{\sigma_{\mathrm{I}}} \quad (5)$$

$$f_{\mathrm{fs}}(p) = \frac{p}{\sigma_{\mathrm{D}}} \quad (6)$$

In (2) – (6), $\bar{p}$ represents the mean pixel value and σ represents the standard deviation of pixel values. The subscripts 'I' and 'D' respectively denote statistics calculated over pixels from a single image (I) and statistics calculated over the entire dataset (D).

In this study, each of the above pre-processing operations is performed on input data.

3) Deep learning algorithms: As discussed in Section I, DL is a subset of machine learning and involves mimicking the human brain. DL algorithms follow a common structure, to the extent that they adopt a layered architecture with multiple nodes at each layer. The DL algorithms for this study are designed to perform a binary classification in determining whether the eye state is 'opened' or 'closed'. A brief overview of the different DL algorithms implemented in this study for image classification is provided below.

a) Convolutional neural network (CNN): The CNN is the most popular artificial neural network (at the time of writing). There are typically three classes of layers in a CNN: convolution layers, pooling layers and fully-connected layers [16, 40]. Fig. 3, re-produced from [41], illustrates the layout of these layers.

Convolution and pooling layers work together to perform feature extraction from the input image [16, 40]. First, input data representing pixels of an image is multiplying the kernel filters of a convolution layer to generate feature maps. Thereafter, a pooling layer is used to group features together and reduce the size of the feature maps. Pooling features together improves the computation time of the DL algorithm [16].

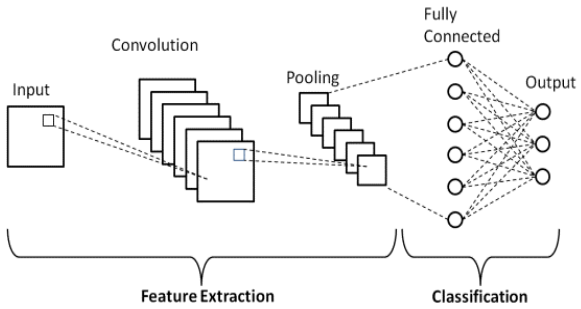

Fig. 3. Basic CNN architecture [41]

TABLE II. ARCHITECTURAL LAYERS OF CNN-C1

Layer Number	Properties
1	Convolutional Layer. 32 nodes, same padding, 3×3 kernel
2	Max pooling layer
3	Convolutional Layer. 32 nodes, same padding, 3×3 kernel
4	Max pooling layer
5	Convolutional Layer. 64 nodes, same padding, 3×3 kernel
6	Max pooling layer
7	Flatten layer
8	Dense layer with 128 units
9	Dense Softmax output layer with two units

TABLE III. ARCHITECTURAL LAYERS OF CNN-C2

Layer Number	Properties
1	Convolutional Layer. 8 nodes, same padding, 3×3 kernel
2	Average pooling layer
3	Convolutional Layer. 16 nodes, same padding, 3×3 kernel
4	Average pooling layer
5	Flatten layer
6	Dense layer with 120 units
7	Dense layer with 84 units
8	Dense Softmax output layer with two units

The processed feature maps are then fed into one or more fully-connected layers. The final layer is referred to as the output layer, and any fully-connected layers between the pooling layer and the output layer are referred to as hidden layers. Each node in a fully-connected layer performs a mathematical operation on its input data using an activation function. These activation functions are selected to map inputs to suitable outputs and perform classification [42].

Two different CNN model configurations were investigated in this study. For brevity, they are referred to as CNN-C1 and CNN-C2. Their respective architectures are shown in Table II and Table III.

Table II describes the first CNN architecture used in this study. These layers are arranged sequentially in a linear stack [43]. The first two convolution layers in this model have 32 nodes each, which are responsible for learning multiple spatial patterns and features from the input image [44]. The last convolution layer 64 nodes. A 3×3 kernel filter is used in each convolution layer, to generate the feature maps. Each convolution layer applied same padding to the input image, which enabled the image to get completely covered by the kernel filter, to generate a feature map [45]. Furthermore, each convolution layer was followed by a pooling layer that applies a maximum filter (max pooling). Once the convolution was completed, the data was then passed to the flatten layer to flatten the multi-dimensional feature map into one dimension [46]. This single dimensional array was then forwarded into the dense layer of the network. A dense layer of 128 units is then used to perform the image classification, using the output from the convolution layers [47]. The last layer of this network was a two-unit output layer which made use of a softmax activation function that calculated the probabilities of each class [48]. There are only two units used in the output layer, because these models are binary classifiers, with predictions made for only two class labels. The output produced by the softmax layer, is represented in the form of a vector, which contains the probabilities of each class, for every sample data

In addition, a Rectified Linear Unit (ReLU) activation function was added to each convolution layer and dense layer, to ensure no negative values were passed to the subsequent layers [16]. The ReLU activation function is given by:

$$f_{\mathrm{ReLU}}(x) = \max(0, x) \qquad (7)$$

In (7), x refers to the input data to the activation function.

Table III describes the second CNN configuration used in this study, which also consists of sequential layers. This configuration uses fewer convolution layers than CNN-C1, but more fully-connected layers when performing classification. CNN-C2 also applies an averaging filter in the pooling layers (average pooling). A with CNN-C1, a ReLU activation function was added to each convolutional layer and dense layer, to ensure no negative values propagated through the network.

b) Visual geometry group (VGG) networks 16 and 19: The VGG have conducted extensive research into DL algorithms for image classifications that improve upon the traditional CNN [49]. The two VGG algorithms chosen were VGG16 [50] and VGG19 [51]. The VGG16 model consists of 13 convolution layers, five max pooling layers, two fully-connected layers and one softmax activation layer at the output [50]. The VGG19 model comprises of 16 convolution layers, five max pooling layers, three fully-connected layers and one softmax activation layer at the output [51].

The VGG19 and VGG16 models used in this study were built using the Keras pre-trained VGG library. As with CNN-C1 and CNN-C2, the output layer was configured to have two units with a softmax output representing the probability on an image falling into either classification.

c) Generative adversarial network (GAN): GANs are a class of DL algorithms that has been applied to image classification problems [52]. The structure of a GAN, shown in Fig. 4 [53], comprises of two sub-neural networks: a generator network and a discriminator network.

During training, both the generator and the discriminator learn concurrently. The function of a generator network is to produce new, artificial instances of data/images from the input features [52]. This is a form of data augmentation that occurs within the network architecture. The artificial images output from the generator network are evaluated by the discriminator to determine whether they adequately resemble images from the true training dataset. Back-propagation is then used to iteratively train the generator. Generator networks are typically seeded with randomized noise data.

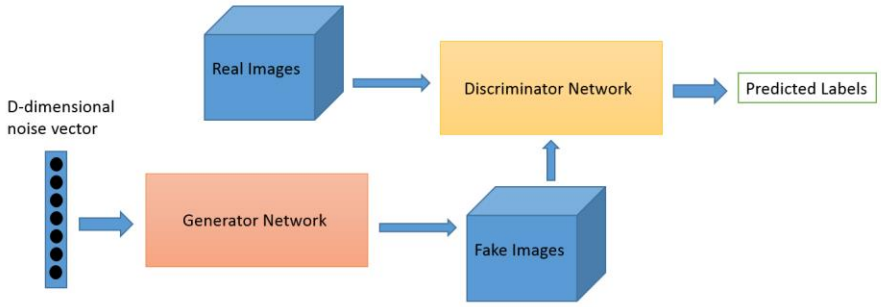

Fig. 4. Structure of a GAN [53]

TABLE IV. ARCHITECTURAL LAYERS OF GAN (DISCRIMINATOR)

Layer Number	Properties
1	Convolutional Layer. 128 nodes, same padding, 3×3 kernel.
2	Max pooling layer
3	Convolutional Layer. 128 nodes, same padding, 3×3 kernel.
4	Max pooling layer
5	Convolutional Layer. 128 nodes, same padding, 3×3 kernel.
6	Max pooling layer
7	Flatten layer
8	Dense Softmax output layer with two units

The discriminator network is trained with images from both the actual dataset and the artificial images produced by the generator. When using a GAN, the discriminator is the final trained model that is tested and deployed in a system.

In the design of a GAN, the discriminator is often a CNN model, and the generator is often a de-convolutional neural network.

The GAN models in this study were built with the architectural layers described in Table IV. There were three convolutional layers used in this network with each layer having 128 nodes. Each convolutional layer was followed by a pooling layer to perform down-sampling. The data was then flattened and passed to a two-unit softmax output layer, where the output prediction was produced. The GAN models deployed a Leaky ReLU activation function, as described by (8), which was added to each down-sampling layer and dense layer. The Leaky ReLU activation function dampens the effect of negative values [54], but does not force them to zero like the standard ReLU function in (7).

$$f_{\text{Leaky ReLU}}(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (8)$$

TABLE V. ARCHITECTURAL LAYERS OF MLP

Layer Number	Properties
1	Flatten Layer
2	Dense layer with 128 nodes.
3	Dense Softmax output layer with two nodes.

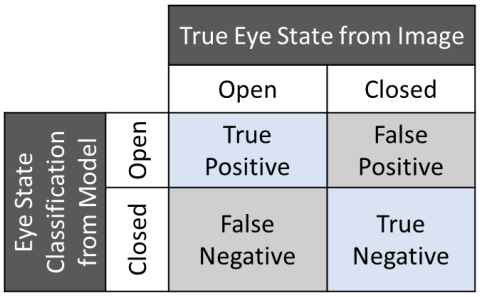

Fig. 5. Definitions matrix of model output classifications

d) Multi-layer perceptron (MLP): The MLP is a more basic DL architecture than those derived from the CNN, as it only consists of fully-connected layers [55, 56]. The typical structure of an MLP consists of an input layer, an output later and at least one hidden layer between the input and output layers. As such, the operation of the MLP is the same as classification stage of a CNN. As a result, MLPs require data to be flattened at the input layer.

The MLP models in this study were built according to the architectural layers described in Table V. The ReLU activation function was implemented in the hidden layer.

4) Model evaluation: When analysing model performance, this study considers a range of metrics collectively to provide a holistic evaluation of performance. The following performance metrics were used to evaluate the DL models: accuracy score, precision, negative class precision, recall, specificity and F1-score. These metrics are defined in (9) – (14), in terms of the number of true positive classifications (T^+), the number of true negative classifications (T^-), the number of false positive classifications (F^+) and the number of false negative classifications (F^-). These output classifications relate true eye state (based on the known label associated with an image) to the detected eye state (based on the output of the model). The definitions of the different output classifications are visually represented in Fig. 5.

a) Accuracy score: The accuracy score is a measure of how many correct predictions were made by the classifier, out of all the predictions made [57, 58]. This is hence the percentage of true output classifications with respect to all output classifications, and is mathematically described as:

$$\text{Accuracy Score} = \frac{T^+ + T^-}{T^+ + T^- + F^+ + F^-} \quad (9)$$

b) Recall and specificity: Recall defines how well the model can correctly classify positive outcomes [58, 59]. In the context of this study, recall indicates how many images of open eyes were correctly classified by the model. In addition, for a balanced evaluation of the predictions made for both class labels, the specificity metric was also used. Specificity indicates how well the model can correctly classify negative outcomes [58]. In the context of this study, it indicates how many images of closed eyes were correctly classified by the model. For the problem of driver drowsiness detection, being able to correctly identify when the driver's eyes are closed is of equal importance than identifying when the eye state is open. The mathematical definitions of recall and specificity are given in (10) and (11), respectively.

$$\text{Recall} = \frac{T^+}{T^+ + F^-} \quad (10)$$

$$\text{Specificity} = \frac{T^-}{T^- + F^+} \quad (11)$$

c) Precision: Precision defines how well a model can make classify positive outputs [60]. In the context of this study, this indicates the percentage of correct open eye state classifications from all open eye state classifications, as shown by (12). Similarly, the negative class precision represents the percentage of correct closed eye state classifications from all closed eye state classifications. The formula for negative class precisions is presented in (13).

$$\text{Precision} = \frac{T^+}{T^+ + F^+} \quad (12)$$

$$\text{Neg. Class Precision} = \frac{T^-}{T^- + F^-} \quad (13)$$

d) F1-Score: The F1-Score represents a weighted average between precision and recall and is hence considered the most appropriate measure of model performance in some literature [57, 61]. Equation (14) presents the mathematical formula to calculate F1-Score [61, 62].

$$\text{F1–Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

IV. Results and Discussion

This section presents and analyses the effects of data augmentation on model performance. Pre-treatment and post-treatment results are presented in Table VI and Table VII, and their descriptive statistics are presented in Table VIII. The change in performance metrics due to treatment is presented in Table IX. While results for all performance metrics are presented, the main analysis focuses mostly on accuracy and F1-score, as the latter provides insight into the underlying precision and recall.

In the analysis carried out, the authors first confirmed that the DA techniques adopted in this study have improved the performance of the DL models that were investigated. Fig. 6 presents a box-and-whisker diagram of the statistical distribution of all evaluation metrics considered; and compares pre-treatment results with post-treatment results. From the results in Fig. 6, Table VII, Table VIII and Table IX, the following observations and interpretations were made:

1) The post-treatment mean and median values of all evaluation metrics are higher than the pre-treatment values (Table VIII and Table IX). This indicates that the average performance of all DL models studied improved due to the DA techniques applied. The average improvement of the most conclusive metrics, accuracy and F1-score, were +6.1% and +6.8% respectively.

2) The interquartile ranges (IQRs) and standard deviations of post-treatment results were less than for pre-treatment results. In terms of the most conclusive metrics, accuracy and F1-Score, the IQR of both metrics decreased from 13% to 3%. The standard deviation of accuracy scores decreased from 0.17 to 0.12. Similarly, the standard deviation of F1-Scores decreased from 0.20 to 0.14. This indicates that there is less variability in the expected post-treatment performance of all DL algorithms.

3) Outliers were noted in the results, which are clearly illustrated in Fig. 6. These arose from the VGG16 and VGG19 models which were trained on Dataset 1 and displayed inferior performance to the other models studied. Upon investigation, this has been attributed to the dimensionality mismatch between Dataset 1 images (96×96 pixels) and the input dimensions defined by the VGG16 and VGG19 architectures (224×224). While the application of DA techniques has shown the greatest improvement to these models, the post-treatment performance is still low compared to the other models studied. It is thus concluded that the VGG models are not suitable for Dataset 1, and in practice, should not be used with low-resolution cameras that produce smaller video frames/images.

TABLE VI. PRE-TREATMENT AND POST-TREATMENT EVALUATION METRICS (ACCURACY, RECALL AND SPECIFICITY)

Algorithm	Dataset	Accuracy Score		Recall		Specificity	
		Pre-treatment	*Post-treatment*	*Pre-treatment*	*Post-treatment*	*Pre-treatment*	*Post-treatment*
CNN-C1	1	93.8%	97.6%	100%	100%	88%	96%
	2	96.5%	97.9%	94%	99%	99%	97%
VGG19	1	47.6%	60.8%	24%	28%	71%	93%
	2	82.4%	95.8%	65%	94%	100%	98%
VGG16	1	50.4%	67.8%	10%	44%	91%	91%
	2	93.1%	95.2%	88%	92%	98%	99%
CNN-C2	1	91.4%	96.2%	100%	97%	83%	97%
	2	95.9%	96.9%	93%	94%	99%	100%
GAN	1	95.4%	97.0%	100%	100%	91%	94%
	2	96.2%	97.2%	97%	97%	95%	98%
MLP	1	91.8%	93.8%	100%	93%	84%	94%
	2	82.1%	93.4%	97%	90%	67%	97%

TABLE VII. PRE-TREATMENT AND POST-TREATMENT EVALUATION METRICS (PRECISION, NEGATIVE CLASS PRECISION AND F1-SCORE)

Algorithm	Dataset	Precision		Neg. Class Precision		F1-Score	
		Pre-treatment	*Post-treatment*	*Pre-treatment*	*Post-treatment*	*Pre-treatment*	*Post-treatment*
CNN-C1	1	89%	96%	100%	100%	94%	98%
	2	99%	97%	95%	99%	97%	98%
VGG19	1	45%	81%	48%	57%	31%	61%
	2	100%	98%	74%	94%	82%	96%
VGG16	1	52%	83%	50%	62%	50%	58%
	2	98%	99%	89%	92%	93%	95%
CNN-C2	1	85%	96%	100%	97%	91%	96%
	2	99%	100%	93%	94%	96%	97%
GAN	1	92%	94%	100%	100%	95%	97%
	2	95%	98%	97%	97%	96%	97%
MLP	1	86%	94%	100%	93%	92%	94%
	2	75%	96%	96%	91%	82%	93%

TABLE VIII. DESCRIPTIVE STATISTICS OF EVALUATION METRICS

`		Accuracy Score	Recall	Specificity	Precision	Neg. Class Precision	F1-Score
Mean	Pre-Treatment	85%	81%	89%	85%	87%	83%
	Post-Treatment	91%	86%	96%	94%	90%	90%
First Quartile	Pre-Treatment	82%	82%	84%	83%	85%	82%
	Post-Treatment	94%	92%	94%	94%	92%	94%
Median	Pre-Treatment	92%	96%	91%	91%	96%	93%
	Post-Treatment	96%	94%	97%	96%	94%	96%
Third Quartile	Pre-Treatment	96%	100%	98%	98%	100%	95%
	Post-Treatment	97%	98%	98%	98%	98%	97%
IQR	Pre-Treatment	13%	18%	15%	16%	15%	13%
	Post-Treatment	3%	6%	4%	4%	6%	3%
Standard Deviation	Pre-Treatment	0.17	0.30	0.10	0.18	0.18	0.20
	Post-Treatment	0.12	0.23	0.03	0.06	0.14	0.14

TABLE IX. EFFECT OF TREATMENT ON EVALUATION METRICS

Algorithm	Dataset	Accuracy Change	Recall Change	Specificity Change	Precision Change	Neg. Class Precision Change	F1-Score Change
CNN-C1	1	+3.8%	0%	+8%	+7%	0%	+4%
	2	+1.4%	+5%	-2%	-2%	+4%	+1%
VGG19	1	+13.2%	+4%	+22%	+36%	+9%	+30%
	2	+13.4%	+29%	-2%	-2%	+20%	+14%
VGG16	1	+17.4%	+34%	0%	+31%	+12%	+8%
	2	+2.1%	+4%	+1%	+1%	+3%	+2%
CNN-C2	1	+4.8%	-3%	+14%	+11%	-3%	+5%
	2	+1.0%	+1%	+1%	+1%	+1%	+1%
GAN	1	+1.6%	0%	+3%	+2%	0%	+2%
	2	+1.0%	0%	+3%	+3%	0%	+1%
MLP	1	+2.0%	-7%	+10%	+8%	-7%	+2%
	2	+11.4%	-7%	+30%	+21%	-5%	+11%
Average		***+6.1%***	***+5.0%***	***+7.3%***	***+9.8%***	***+2.8%***	***+6.8%***

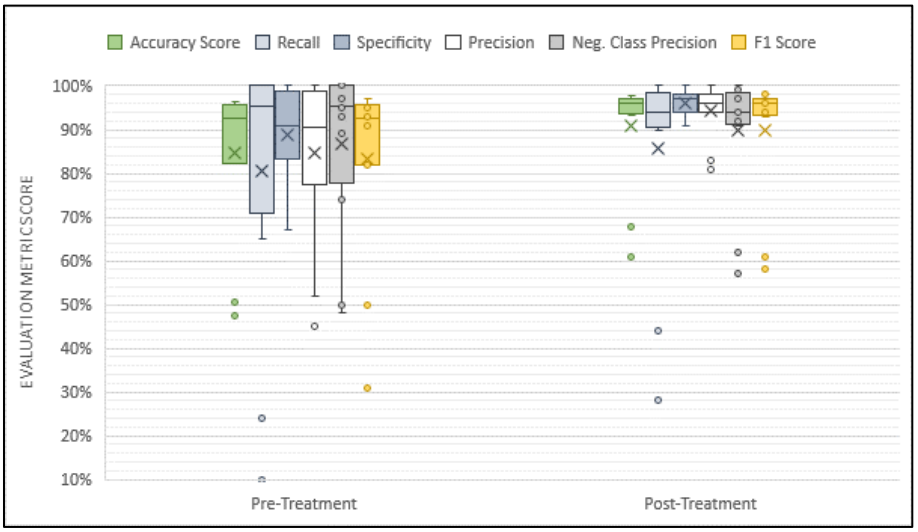

Fig. 6. Statistical distribution of pre-treatment and post-treatment evaluation metrics

TABLE X. EFFECT OF TREATMENT ON EVALUATION METRICS – OUTLIERS REMOVED

Algorithm	Dataset	Accuracy Change	Recall Change	Specificity Change	Precision Change	Neg. Class Precision Change	F1 Score Change
CNN-C1	1	+3.8%	0%	+8%	+7%	0%	+4%
	2	+1.4%	+5%	-2%	-2%	+4%	+1%
VGG19	2	+13.4%	+29%	-2%	-2%	+20%	+14%
VGG16	2	+2.1%	+4%	+1%	+1%	+3%	+2%
CNN-C2	1	+4.8%	-3%	+14%	+11%	-3%	+5%
	2	+1.0%	+1%	+1%	+1%	+1%	+1%
GAN	1	+1.6%	0%	+3%	+2%	0%	+2%
	2	+1.0%	0%	+3%	+3%	0%	+1%
MLP	1	+2.0%	-7%	+10%	+8%	-7%	+2%
	2	+11.4%	-7%	+30%	+21%	-5%	+11%
Average		**+4.3%**	**+2.2%**	**+6.6%**	**+5.0%**	**+1.3%**	**+4.3%**

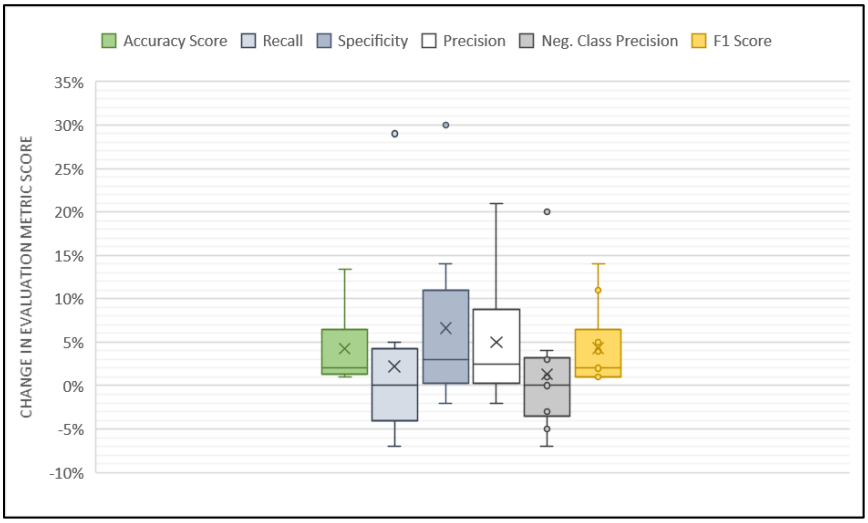

Fig. 7. Statistical distribution of change in evaluation metric scores

Having confirmed the hypothesis that the DA techniques that were applied have improved the performance of the DL models studied, the next step was to attempt to quantify the extent of this improvement. The VGG16 and VGG19 models trained on Dataset 1 were excluded from this analysis due to their poor performance, as discussed previously. Table X presents the change in evaluation metrics due to the application of DA with these models removed. The statistical distribution of the data presented in Table X is illustrated in Fig. 7.

When analyzing the results, the following was observed:

1) A few instances were observed where applying DA treatment caused a reduction in individual evaluation metrics (recall, precision, specificity and negative class precision), as indicated by shaded backgrounds within Table X. However, despite this, the F1-Score increased for all models, indicating that these performance reductions were compensated for. The average increase in both accuracy and F1-Score was +4.3%, and the median increase in each of these metrics were +2.1% (accuracy) and +2.0% (F1-Score).

2) The box-and-whisker diagrams in Fig. 7 indicated that there is significantly more variability for recall, specificity, precision and negative-class precision than for accuracy and F1-Scores. As such, attempts at quantifying the expected improvement in DL model performance using the methods in this study can only reasonable be performed for accuracy and F1-Score. However, these are the most conclusive metrics to evaluate the DL models studied.

3) By analyzing the distribution of the change in accuracy and F1-Scores, it was observed that the data for these evaluation metrics was positively skewed. This resulted from the high pre-treatment accuracy scores and F1-Scores of some of the DL models studied, where there was not much room for improvement without over-fitting the model to the training dataset.

Prompted by the final observation listed above, the final analysis investigated the relationship between the change in evaluation metric scores and pre-treatment metric scores. The scatterplot presented in Fig. 8 illustrates this relationship, using data from Table VI, Table VII and Table X and excluding the outlier results resulting from the VGG16 and VGG19 models that were trained on Dataset 1. The trend lines show that all evaluation metrics exhibited a strong negative correlation, indicated by the R^2 values of the correlation trend lines ($R^2 > 0.7$ for all evaluation metrics). From this, it is concluded that the DA techniques under study have a marginal improvement when applied to DL models that already exhibit strong performance, but are much more powerful in enhancing weaker-performing DL models. From Fig. 8, an improvement of $\geqslant$ +5% to an evaluation metric occurs when the pre-treatment value of the metric is $\leqslant$ 90%. This indicates the type of DL models for driver drowsiness detection that will benefit most from the DA techniques presented in this study, and is recommended to developers as a design guideline when considering the implementation of the DA techniques presented in this paper.

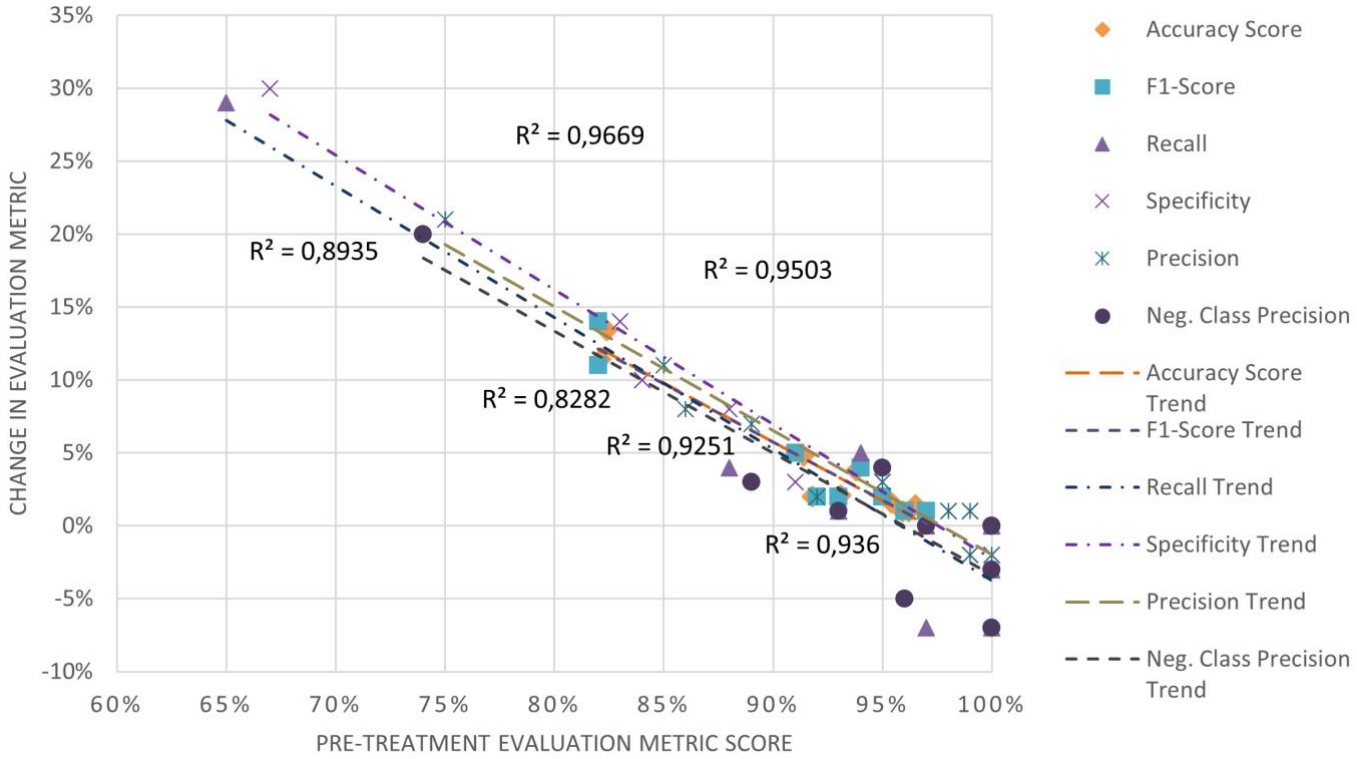

Fig. 8. Relationship between pre-treatment evaluation metric score and change in evaluation metric score

The results confirm that by modelling real-world scenarios using the data augmentation techniques described in Section II.B.2, it is possible to train more robust deep learning algorithms that perform driver drowsiness detection. With respect to implementation of driver drowsiness detection systems, the deep learning model development and training would be performed before the model is deployed in the driver drowsiness detection system hardware.

V. Conclusion

Many road accidents are caused by driver drowsiness. Previous studies have considered applying deep learning techniques to detect driver drowsiness and improve road traffic safety. In practically testing their systems, many previous studies have indicated that real-world scenarios such as unfavourable ambient lighting and movement of the driver while driving cause inaccuracies when detecting driver drowsiness.

In this study, the authors focussed on the deep learning algorithms that determine driver drowsiness based on the eye state of the driver. It was hypothesised that by modelling the real-world scenarios and using data augmentation techniques on a standardised image dataset, the performance of the DL models would improve. This study considered two different datasets, six different DL models: two CNN variations (CNN-C1 and CNN-C2), two architectures designed by the VGG (VGG16 and VGG19), a GAN and an MLP.

The performance of the DL models was evaluated primarily using accuracy and F1-Score, although other metrics such as precision, recall, specificity and negative class precision were also considered. In analyzing the results in aggregation, improvements across all metrics were noted. The average improvement in accuracy across all DL models was +6.1% and the average improvement in F1-Score was 6.8%, and the variability in model performance was reduced. However, there were some challenges noted when training the VGG models. These models trained on low-resolution images, exhibited poor performance and distorted these results. A more realistic indication of the benefits of DA for the DL models studied was obtained by excluding these outliers, yielding an average improvement of +4.3% for both accuracy and F1-Score.

The results further indicated that the extent to which the DA techniques studied improve DL model performance is strongly correlated with the pre-treatment DL model performance. From the analysis conducted, the data augmentation techniques presented are best suited for improving models with accuracy and F1-Scores $\leqslant$ 90% - although they are applicable to any DL model for driver drowsiness detection.

It was thus concluded that the use of DA techniques improves the performance of DL models for driver drowsiness detection under the isolated conditions of this study. However, since the conditions of this study focussed on testing the DL models on images from datasets, rather than testing being done on captured data from a real-world driver drowsiness detection system, this opens the possibility for future research. Future works should look at implementing the trained DL models proposed in this study in practical driver drowsiness detection systems to validate these results.

Data Availability

The drowsiness detection dataset (Dataset 1) is available online at: https://www.kaggle.com/prasadvpatil/mrl-dataset.

The yawn eye dataset (Dataset 2) is available online at: https://www.kaggle.com/serenaraju/yawn-eye-dataset-new.

Funding

All funding in support of this research was provided by the Durban University of Technology.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgment

The authors acknowledge the support of the wider Faculty of Accounting and Informatics at the Durban University of Technology, and Mr Shaldon Wade Naidoo for his assistance.

References

[1] D. Machetele and K. Yessoufou, "A Decade Long Slowdown in Road Crashes and Inherent Consequences Predicted for South Africa," (in English), Frontiers in Future Transportation, Original Research vol. 2, 2021-November-03 2021, doi: 10.3389/ffutr.2021.760640.

[2] T. Verster and E. Fourie, "The good, the bad and the ugly of South African fatal road accidents," South African Journal of Science, vol. 114, pp. 63-69, 2018. [Online]. Available: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S0038-23532018000400017&nrm=iso.

[3] M. Mokoatle, V. Marivate, and M. E. Bukohwo, "Predicting Road Traffic Accident Severity using Accident Report Data in South Africa," presented at the Proceedings of the 20th Annual International Conference on Digital Government Research, Dubai, United Arab Emirates, 2019. [Online]. Available: https://doi.org/10.1145/3325112.3325211.

[4] L. Weisberg. "The Long-Term Effects of Car Accident Injuries." Weisberg Cummings, P.C. Emplyment Law Attorneyshttps://www.weisbergcummings.com/long-term-effects-car-accident-injuries/ (accessed 10 November, 2022).

[5] A. B. Boye, "The Effect of Road Traffic Delay on Supply Chain Performance of Manufacturing Firms in Lagos State, Nigeria," International Journal of Management Technology, vol. 5, no. 3, pp. 9-20, 2018. [Online]. Available: https://www.eajournals.org/journals/international-journal-of-management-technology-ijmt/vol-5-issue-3-september-2018/the-effect-of-road-traffic-delay-on-supply-chain-performance-of-manufacturing-firms-in-lagos-state-nigeria/.

[6] MiWay. "The truth about truck accidents on SA roads." https://www.miway.co.za/blog/business-insurance/the-truth-about-truck-accidents-on-sa-roads (accessed 10 November, 2022).

[7] Dovetail. "6 Road Freight Challenges in South Africa." https://dovetail.co.za/6-road-freight-challenges-in-south-africa/ (accessed 10 November, 2022).

[8] Daily News Reporter. "SA has the world's poorest road safety records - WHO report." IOL News. https://www.iol.co.za/thepost/news/sa-has-the-worlds-poorest-road-safety-records-who-report-18631896 (accessed 10 November, 2022).

[9] L. Rondganger. "South Africa's roads deaths are a 'national crisis'." IOL News. https://www.iol.co.za/news/south-africa/kwazulu-natal/south-africas-roads-deaths-are-a-national-crisis-cefc54fe-bafe-45c6-b0d9-f7c4b1ce7ea8 (accessed 10 November, 2022).

[10] B. Alshaqaqi, A. S. Baquhaizel, M. E. A. Ouis, M. Boumehed, A. Ouamri, and M. Keche, "Driver drowsiness detection system," in 2013

8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA), 12-15 May 2013 2013, pp. 151-155, doi: 10.1109/WoSSPA.2013.6602353.

[11] L. J. Hurwitz. "Driver Fatigue: A Leading Cause of Accidents and Death in the Transportation Industry." Segal McCambridge. https://www.segalmccambridge.com/blog/driver-fatigue-a-leading-cause-of-accidents-and-death-in-the-transportation-industry/ (accessed 10 November, 2022).

[12] M. Ramzan, H. U. Khan, S. M. Awan, A. Ismail, M. Ilyas, and A. Mahmood, "A Survey on State-of-the-Art Drowsiness Detection Techniques," IEEE Access, vol. 7, pp. 61904-61919, 2019, doi: 10.1109/ACCESS.2019.2914373.

[13] R. Jabbar, K. Al-Khalifa, M. Kharbeche, W. Alhajyaseen, M. Jafari, and S. Jiang, "Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques," Procedia Computer Science, vol. 130, pp. 400-407, 2018/01/01/ 2018, doi: https://doi.org/10.1016/j.procs.2018.04.060.

[14] R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, "Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application," in 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), 2-5 Feb. 2020 2020, pp. 237-242, doi: 10.1109/ICIoT48696.2020.9089484.

[15] S. S. Sharan, R. Viji, R. Pradeep, and V. Sajith, "Driver Fatigue Detection Based On Eye State Recognition Using Convolutional Neural Network," in 2019 International Conference on Communication and Electronics Systems (ICCES), 17-19 July 2019 2019, pp. 2057-2063, doi: 10.1109/ICCES45898.2019.9002215.

[16] V. R. R. Chirra, S. R. Uyyala, and V. K. K. Kolli, "Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State," Revue d'Intelligence Artificielle, vol. 33, no. 6, pp. 461-466, 2019, doi: 10.18280/ria.330609

[17] M. Hashemi, A. Mirrashid, and A. B. Shirazi, "Driver Safety Development: Real-Time Driver Drowsiness Detection System Based on Convolutional Neural Network," SN Computer Science, vol. 1, no. 5, p. 289, 2020/08/31 2020, doi: 10.1007/s42979-020-00306-9.

[18] F. Zhang, J. Su, L. Geng, and Z. Xiao, "Driver Fatigue Detection Based on Eye State Recognition," in 2017 International Conference on Machine Vision and Information Technology (CMVIT), 17-19 Feb. 2017 2017, pp. 105-110, doi: 10.1109/CMVIT.2017.25.

[19] W. Kongcharoen, S. Nuchitprasitchai, Y. Nilsiam, and J. M. Pearce, "Real-Time Eye State Detection System for Driver Drowsiness Using Convolutional Neural Network," in 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 24-27 June 2020 2020, pp. 551-554, doi: 10.1109/ECTI-CON49241.2020.9158265.

[20] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," WSEAS Transactions on Circuits and Systems, vol. 8, no. 7, pp. 579-588, 2009, doi: 10.5555/1639537.1639542.

[21] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," in 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), 10-12 March 2017 2017, pp. 721-724, doi: 10.1109/ICBDA.2017.8078730.

[22] R. Seetharaman, S. Sridhar, and S. Mootha, "Detection and State Analysis of Drowsiness using Multitask Learning with Neural Networks," in 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), 21-23 Oct. 2020 2020, pp. 1-8, doi: 10.1109/ICDS50568.2020.9268740.

[23] T. Vesselenyi, S. Moca, A. Rus, T. Mitran, and B. Tătaru, "Driver drowsiness detection using ANN image processing," IOP Conference Series: Materials Science and Engineering, vol. 252, no. 1, p. 012097, 2017/10/01 2017, doi: 10.1088/1757-899X/252/1/012097.

[24] A. Ghourabi, H. Ghazouani, and W. Barhoumi, "Driver Drowsiness Detection Based on Joint Monitoring of Yawning, Blinking and Nodding," in 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), 3-5 Sept. 2020 2020, pp. 407-414, doi: 10.1109/ICCP51029.2020.9266160.

[25] P. Bajaj, R. Ray, S. Shedge, S. Jaikar, and P. More, "Synchronous System for Driver Drowsiness Detection Using Convolutional Neural Network, Computer Vision and Android Technology," in 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 19-20 March 2021 2021, vol. 1, pp. 340-346, doi: 10.1109/ICACCS51430.2021.9441670.

[26] H. Ahuja, S. Saurav, S. Srivastava, and C. Shekhar, "Driver Drowsiness Detection using Knowledge Distillation Technique for Real Time Scenarios," in 2020 IEEE 17th India Council International Conference (INDICON), 10-13 Dec. 2020 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342263.

[27] M. Ngxande, J. R. Tapamo, and M. Burke, "Bias Remediation in Driver Drowsiness Detection Systems Using Generative Adversarial Networks," IEEE Access, vol. 8, pp. 55592-55601, 2020, doi: 10.1109/ACCESS.2020.2981912.

[28] P. Chen, S. Liu, H. Zhao, and J. Jia, "Gridmask data augmentation," arXiv preprint arXiv:2001.04086, 2020.

[29] H. J. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," arXiv preprint arXiv:2007.07588, 2020.

[30] P. V. Patil. Drowsiness Detection Dataset, Kaggle. [Online]. Available: https://www.kaggle.com/prasadvpatil/mrl-dataset

[31] S. Raju. yawn_eye_dataset_new, Kaggle. [Online]. Available: https://www.kaggle.com/serenaraju/yawn-eye-dataset-new

[32] A. Takimoglu. "What Is Data Augmentation? Techniques, Benefit & Examples." AIMultiple. https://research.aimultiple.com/data-augmentation/ (accessed 8 November, 2022).

[33] N. Ketkar, "Introduction to Keras," in Deep Learning with Python: Springer, 2017, pp. 97-111.

[34] A. Bhandari. "Image Augmentation on the Fly Using Keras Imagedatagenerator." Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/ (accessed.

[35] F. Chollet. "Building powerful image classification models using very little data." The Keras Blog. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html (accessed 8 November, 2022).

[36] J. Brownlee. "How to Normalize, Center, and Standardize Image Pixels in Keras." Machine Learning Mastery. https://machinelearningmastery.com/how-to-normalize-center-and-standardize-images-with-the-imagedatagenerator-in-keras/ (accessed 10 November, 2022).

[37] J. Brownlee. "A Gentle Introduction to Batch Normalization for Deep Neural Networks." Machine Learning Mastery. https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/2020 (accessed 10 November, 2022).

[38] M. Alam. "Data Normalization in Machine Learning." Towards Data Science. https://towardsdatascience.com/data-normalization-in-machine-learning-395fdec69d02 (accessed 7 November, 2022).

[39] TheAILearner. "Data Augmentation with Keras Imagedatagenerator." TheAILearner. https://theailearner.com/2019/07/06/data-augmentation-with-keras-imagedatagenerator/ (accessed 7 November, 2022).

[40] S. Albawi, T. A. Mohammed, and S. Al-Azawi, "Understanding of a convolutional neural network," in 2017 International Conference on Engineering and Technology (ICET), 21-23 Aug. 2017 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[41] M. K. Gurucharan. "Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network." upGrad Education Private Limited. https://www.upgrad.com/blog/basic-cnn-architecture/ (accessed 1 December, 2021).

[42] Y. Wang, Y. Li, Y. Song, and X. Rong, "The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition," Applied Sciences, vol. 10, no. 5, p. 1897, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/5/1897.

[43] F. Chollet. "The Sequential Model." Keras.io. https://keras.io/guides/sequential_model/ (accessed 8 November, 2022).

[44] R. Keshari, M. Vatsa, R. Singh, and A. Noore, "Learning Structure and Strength of CNN Filters for Small Sample Size Training," in 2018

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18-23 June 2018 2018, pp. 9349-9358, doi: 10.1109/CVPR.2018.00974.

[45] A. Wiranata, S. A. Wibowo, R. Patmasari, R. Rahmania, and R. Mayasari, "Investigation of Padding Schemes for Faster R-CNN on Vehicle Detection," in 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 5-7 Dec. 2018 2018, pp. 208-212, doi: 10.1109/ICCEREC.2018.8712086.

[46] Z. J. Wang et al., "CNN 101: Interactive visual learning for convolutional neural networks," in Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1-7.

[47] G. Dumane. "Introduction to Convolutional Neural Network (CNN) Using Tensorflow." Towards Data Science. https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83 (accessed 8 November, 2022).

[48] S. Sharma, S. Sharma, and A. Athaiya, "Activation Functions in Neural Networks," International Journal of Engineering Applied Sciences and Technology, vol. 4, no. 12, pp. 310-316, 2020.

[49] OpenGenus IQ. "Understanding the VGG19 Architecture." OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/vgg19-architecture/ (accessed 31 October, 2021).

[50] T. J. Perumanoor. "What Is VGG16? — Introduction to VGG16." Great Learning. https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615 (accessed 17 November, 2021).

[51] L. Wen, X. Li, X. Li, and L. Gao, "A New Transfer Learning Based on VGG-19 Network for Fault Diagnosis," in 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD), 6-8 May 2019 2019, pp. 205-209, doi: 10.1109/CSCWD.2019.8791884.

[52] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53-65, 2018, doi: 10.1109/MSP.2017.2765202.

[53] C. Nicholson. "A Beginners Guide to Generative Adversarial Networks (GANs)." Pathmind. https://wiki.pathmind.com/generative-adversarial-network-gan (accessed 4 March, 2022).

[54] A. K. Dubey and V. Jain, "Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions," in Applications of Computing, Automation and Wireless Systems in Electrical Engineering, Singapore, S. Mishra, Y. R. Sood, and A. Tomar, Eds., 2019// 2019: Springer Singapore, pp. 873-880.

[55] H. Zhu, D. Gao, and S. Zhang, "A Perceptron Algorithm for Forest Fire Prediction Based on Wireless Sensor Networks," Journal on Internet of Things, vol. 1, no. 1, 2019, doi: 10.32604/jiot.2019.05897.

[56] J. Brownlee. "Perceptron Algorithm for Classification in Python." Towards Data Science. https://machinelearningmastery.com/perceptron-algorithm-for-classification-in-python/ (accessed 4 April, 2022).

[57] D. R. Govinda and Z. Waheed, "Increasing Trend in Accuracy Score for Machine Learning Algorithms," International Journal of Development and Public Policy, vol. 1, no. 5, pp. 180-182, 10/23 2021. [Online]. Available: https://www.openaccessjournals.eu/index.php/ijdpp/article/view/429.

[58] S. Raschka and V. Mirjalili, Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd, 2019.

[59] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. Cohen, "Revisiting Training Strategies and Generalization Performance in Deep Metric Learning," presented at the Proceedings of the 37th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2020. [Online]. Available: https://proceedings.mlr.press/v119/roth20a.html.

[60] M. Dua, Shakshi, R. Singla, S. Raj, and A. Jangra, "Deep CNN models-based ensemble approach to driver drowsiness detection," Neural Computing and Applications, vol. 33, no. 8, pp. 3155-3168, 2021/04/01 2021, doi: 10.1007/s00521-020-05209-7.

[61] S. Panda and M. Kolhekar, "Feature Selection for Driver Drowsiness Detection," in Proceedings of International Conference on Computational Intelligence and Data Engineering, Singapore, N. Chaki, N. Devarakonda, A. Sarkar, and N. C. Debnath, Eds., 2019// 2019: Springer Singapore, pp. 127-140.

[62] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," BMC Genomics, vol. 21, no. 1, p. 6, 2020/01/02 2020, doi: 10.1186/s12864-019-6413-7.