International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 16, No.1 (Sep-2024)

http://dx.doi.org/10.12785/ijcds/1601106

High-Fidelity Machine Learning Techniques for Driver Drowsiness Detection

Ebenezer Essel[1], Abeer Abdelhamid[2], Mahmoud Darwich[3], Fahmi Khalifa[4], Fred Lacy[5] and Yasser Ismail[5]

[1]*Department of Electrical & Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, USA*
[2]*Electronics and Communications Engineering Dept., Mansoura University, Mansoura 35516, Egypt*
[3]*Department of Mathematics and Computer Science, University of Mount Union, Alliance, Ohio 44601, USA*
[4]*Department of Electrical and Computer Engineering, Morgan State University, Baltimore MD 21251, USA*
[5]*Department of Electrical Engineering, Southern University and A&M College, Baton Rouge, LA 70813, USA*

Received 16 Mar. 2024, Revised 10 Jun. 2024, Accepted 11 Jun. 2024, Published 20 Sep. 2024

Abstract: It is devastating that daily, there is an ample number of car crashes that cause damage to automobiles, onboard passengers get injured, and others tend to lose their lives. Road crashes are fast rising across the globe and have drawn many road safety commissions and concerned individuals to discuss ways to reduce this menacing situation drastically. With the introduction of artificial intelligence and technological advancement, the government and state commissions have beckoned on the various universities and research institutions to develop methods to curb the rise of automobile crashes. Some causes of these crashes include drunk driving and drowsiness, the latter is most prevalent as it occurs to all and sundry. Drowsiness detection can be categorized into three main techniques; behavioral-based, vehicular-based, and physiological-based. In this research, the behavioral-based approach was studied, with significant consideration being the cost of implementation, execution time, and accuracy. This research investigates drowsiness detection using a novel approach that directly utilizes image pixels from facial geometry to enhance accuracy in classification. Unlike prior studies that relied solely on recorded EAR and MOR values from images, our methodology harnesses direct input from facial features, yielding promising results in drowsiness classification. Three machine learning (ML) classifiers were considered: Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forest (RF). A dataset of 1448 images was used for training and testing these classifiers: 70% for training and 30% for testing. Random Forest classifier gave the best accuracy of (92.41%) compared to SVM (90.34%) and Naïve Bayes (69.43%). A deep neural network (VGG16) was used to classify drowsiness, and this gave a high accuracy of 97.20%, which outperformed the traditional machine learning models.

Keywords: Drowsiness Detection; Machine Learning; Automobile Crashes; Artificial Intelligence

1. Introduction

Drowsiness detection is a safety feature used in cars to help avoid crashes caused by drowsy drivers [1]. Numerous detection techniques assess driver tiredness and warn the motorist. Detection methods can be categorized based on behavioral, vehicular, and physiological parameters. Vehicular and behavioral-based approaches are non-invasive. Whereas vehicular-based techniques consider factors like yaw angle, steering wheel behavior, and lane-changing patterns [2], behavioral-based techniques are centralized on the driver's actions, including eye closeness ratio, eye blinking, head movement, and yawning [1]. Physiological approaches monitor the driver's physiological conditions, such as heartbeat, pulse rate, and electrical activity in the brain, and are invasive or intrusive. The geometric properties of different roads make vehicular highly unreliable [1]. The need to purchase different sensors and devices makes physiological-based approaches capital-demanding. Additionally, it is invasive and frequently aggravates and discomforts the driver. Because it is affordable [3], [4] and easy/convenient to apply [4], the behavioral-based method is thus extensively employed. However, how the data is processed, including lighting and illumination, impacts behavioral techniques.

In a study by Chellappa et al. [5], the somatic sensor, temperature sensor, LM-35, and photoplethysmography (PPG) were used to measure the core body temperature and pulse rate. This study employed the integration of behavioral and physiological parameters to detect drowsiness. The Viola-Jones algorithm and Haar cascade classifier were used together with the devices for detection, with an achieved detection accuracy of 80.55%. Awais et al. considered biological parameters such as the heart rate, time-domain, and frequency domain measures extracted

E-mail address: eessel1@lsu.edu, abeer.abdelhamid@mans.edu.eg, darwicma@mountunion.edu fahmi.khalifa@morgan.edu, fred_lacy@subr.edu, yasser_ismail@subr.edu

from electroencephalogram (EEG) and electrocardiogram (ECG) combined with the SVM and k-strongest strengths (kSS) to detect drowsiness [6]. The overall performance of their study was 80%. A summary of some exciting literature on behavioral and vehicular methods is given in [1]. In particular, the study proposed in [7], used the deep learning method and achieved an accuracy of 83.30%. Essel's study in [1] draws the method and main contributions from [8] and the model's applicability to Android devices. Deng and Wu [9] combined the kernelized correlation filters (KCF) and convolutional neural network (CNN) algorithm in detection and called it DriCare with an accuracy of 93.60%. The work by N. Kumar et al. [10] designed a system to detect real-time eye blinking using the Viola-Jones detection and active contour method for yawning. The experiment involved 70 male and 30 female volunteers of different ages and facial characteristics. The experiments were conducted at six different times: (1) Morning (6 AM to 11 AM), (2) Afternoon (11 AM to 2 PM), (3) Critical Time 1 (2 PM to 4 PM), (4) Evening (5 PM to 8 PM), (5) Night (8 PM to Mid Night 3 AM), and (6) Critical Time 2 (Mid Night 3 AM to 6 AM). The result showed an accuracy of 92% for eye detection while mouth detection achieved an accuracy of 88%.

A deep CNN for driver drowsiness detection based on eye state was proposed in [11] . A dataset of 1,200 samples from the video stream and 2850 images were used to train, test, and validate the ML model. The Viola-Jones algorithm was used for face and eye detection. First, the convolution layer in CNN extracted the facial features; then, the SoftMax layer classified images as sleepy or non-sleepy. Two experiments were conducted, and of the two, the first experiment was on 2,850 images (trained 1,200 images; validated 500 images; and tested 1,150 images); the second experiment was conducted on 1,200 video samples. The highest accuracy recorded from the experiment was 96.42%. The authors reiterated that the classification of face detection techniques could be grouped into two, that is, geometric-based techniques and image-based techniques. The geometric extraction method extracts shape and location-related metrics from the eyes and eyebrows. Image-based approaches for face detection use statistical neural networks and linear subspace methods. The study by Latreche et al [12] aimed to identify the most informative brain region for drowsiness detection without sacrificing accuracy. Using a deep learning model and publicly available data, the central region emerged as the most effective single region, achieving a mean accuracy of 73.55%. Combining central and occipital regions further improved accuracy to 75.53%, offering a promising approach to enhance EEG-based drowsiness detection in real-time. However, wearable devices with many electrodes may be uncomfortable and costly.

The work by H. Lamaazi [13], introduces a two-stage driver drowsiness detection system leveraging smart edge computing. By utilizing mobile devices within the vehicle for data capture and analysis without data sharing, privacy concerns are addressed. The proposed framework employs a distributed edge architecture and a data fusion strategy involving facial expression analysis, car path detection, and a two-layer LSTM algorithm, achieving an impressive average accuracy of 97.7% in drowsiness detection. Jebraeily et al. in [14] address the issue of driver drowsiness using a convolutional neural network (CNN) architecture by optimizing a network architecture search mechanism. The research enriches the dataset by extracting videos showcasing drowsy states and labeling the frames accordingly. Through a genetic algorithm, an optimal CNN structure is determined, considering factors like the number of layers and objective function type. Transfer learning is then employed, leveraging the optimized network as a feature extractor, and fine-tuning its fully connected layer for drowsiness detection achieving an accuracy of approximately 99.8%.

Agarkar et al. in the paper [15] propose a camera-based technique leveraging facial cues such as lip movements, eye behavior, and hand gestures, which often accompany yawning. By employing a front camera and Raspberry Pi for image processing, the system continuously monitors the driver's condition. Results indicate the effectiveness of the proposed method achieving an accuracy of 92.75%. In [16], Peddarapu et al. highlight the urgent need for an effective system to detect sleepy driving, emphasizing the importance of early warnings to prevent potential crashes. Utilizing machine learning and computer vision methodologies, that is, exploring behavioral techniques, the system focuses on fatigue indicators such as head pose, facial expressions, and eye tracking for precise drowsiness identification in real-time. Leveraging Raspberry Pi technology ensures stability, adaptability, and cost-efficiency, enhancing accessibility for multipurpose vehicles. By providing timely alerts to drivers, the system plays a crucial role in enhancing road safety. Its readiness to combat drowsy driving marks a significant contribution towards creating safer driving conditions and safeguarding the well-being of drivers.

In this paper, we aim to: (1) Provide a comprehensive review of existing drowsiness detection methodologies, spanning from behavioral cues to advanced machine learning techniques; (2) Compare the performance of traditional ML classifiers with deep learning models in drowsiness detection accuracy; (3) Investigate recent advancements in ML and deep learning methods for driver drowsiness detection; and (4) Contribute insights into the most effective approaches for high-fidelity drowsiness detection in real-world driving scenarios.

The paper is organized as follows, in Section 2, the proposed work is elaborated. Results and discussions are elaborated in Section 3. A conclusion will be drawn in Section 4.

2. Proposed Work

The research outlined in this paper integrates both traditional machine-learning methods and deep-learning ap-

proaches to implement a high-fidelity drowsiness detection algorithm. In our study, the selection of machine learning classifiers (SVM, Random Forest, Naïve Bayes) and the deep neural network (VGG16) was based on rigorous evaluation of their suitability for drowsiness detection tasks. Specific features extracted from facial images, such as color histograms, texture descriptors, and geometric features, were utilized to train the classifiers and fine-tune the neural network. The rationale behind choosing these models and features was to maximize accuracy and generalization performance. A schematic of the pipeline is shown in Fig. 1. The subsequent sections provide comprehensive insights into the utilization of these methodologies.

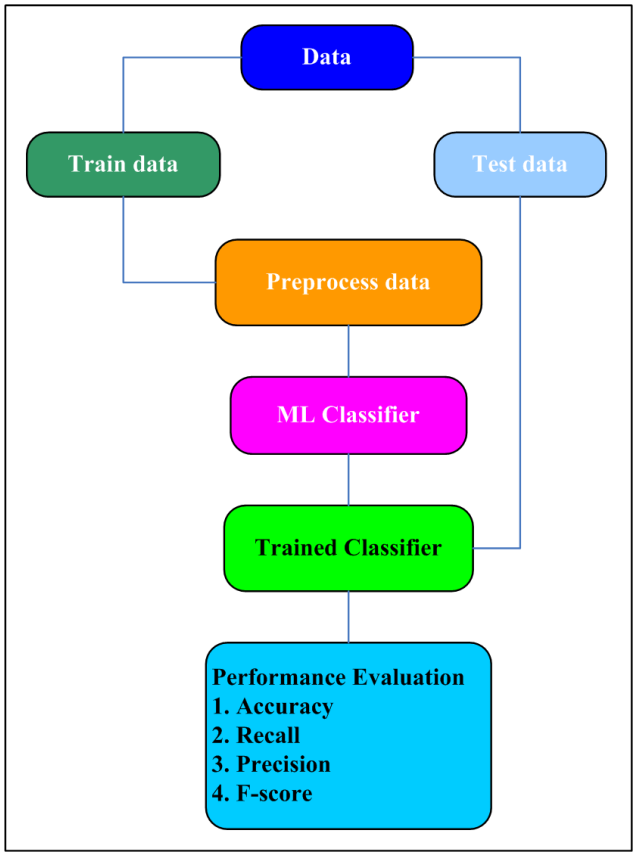

Figure 1. Schematic of the proposed pipeline flow

A. Machine Learning Methods

This study introduces a detection framework utilizing machine learning classifiers to improve effectiveness. The model proposed, along with its implementation stages as depicted in Figure 1–comprising data acquisition, data preprocessing, machine learning training and classification, and performance assessment – each contributes significantly to the system's overall functionality, guaranteeing strong detection capabilities and thorough performance evaluation.

1) Data Acquisition

Illustrated in Figure 1, the model initiates with data gathering, where machine learning algorithms operate on static frames. The dataset, sourced from [17], encompasses participants of various ethnic backgrounds and genders. It includes individuals' images under diverse conditions, such as those with and without glasses, yawning, and not yawning. To ensure the representativeness of the dataset, we included images depicting a diverse range of facial expressions and poses, reflecting real-world scenarios of driver fatigue. Subsequently, the dataset was partitioned into training and testing subsets.

2) Data Preprocessing

During this stage, the training data underwent preprocessing before being fed into the classifier. This involved resizing the images to dimensions of 32×32 to expedite processing and conserve memory resources. Additionally, the labels associated with each image were converted into binary values. The drowsy state was denoted by a binary "zero", while the non-drowsy state was represented as a binary "one". These steps were crucial for enhancing the model's robustness and generalization ability.

3) Machine Learning Classifier and Training Process

Once the training data has undergone preprocessing, it becomes suitable for training the classifiers. The machine learning classifiers employed follow a supervised learning approach. Specifically, three classifiers–SVM, NB, and RF–are *utilized*, with their respective specifications provided below. The selection of these classifiers was based on their suitability for image classification tasks and their comparative performance in previous studies. SVM is known for its effectiveness in separating classes in high-dimensional spaces, while Random Forest is robust against overfitting and works well with diverse features. Naïve Bayes, despite its simplifying assumptions, was included for comparison given its efficiency with large datasets and categorical features.

a) Support Vector Machine (SVM) represents an early classification method in machine learning, originating in the early 1990s and serving as a generalized version of the 'maximal margin classifier'. Initially, SVM was confined to situations with linear boundaries; however, advancements have extended its applicability to diverse datasets. SVM operates on the concept of a hyperplane, where in an n-dimensional space, a hyperplane constitutes a flat affine subspace of dimension (n-1). In two-dimensional (2D) space, the hyperplane equates to a line, while in three dimensional (3D) space, it corresponds to a plane. In 2D, a hyperplane is described by equation (1):

$$mx + ny + c = 0 \quad or \quad mx + p = 0 \tag{1}$$

If a training point lies directly on this equation, it resides on the hyperplane. However, in practice, training observations often fall on either side of the hyperplane, satisfying the inequalities in equation (2):

$$mx + ny + c < 0 \quad or \quad mx + ny + c > 0 \tag{2}$$

The primary objective is to construct a hyperplane that

perfectly segregates the training sets based on their class labels. Subsequently, test data is classified according to the side of the hyperplane it occupies. Despite the possibility of numerous hyperplanes facilitating perfect separation of data, the optimal separating hyperplane is selected based on maximizing the margin away from the training points. This is determined by computing the perpendicular distance from each training point to the given separating hyperplane, with the smallest distance representing the margin. Test observations are then classified based on their relative position to the maximal margin hyperplane.

Support vectors denote the training points closest to the hyperplane, influencing the margin's magnitude. In cases where no separating hyperplane exists, the concept of a soft margin is introduced, allowing some training data to be misclassified to enhance the classifier's performance on other observations. A regularization parameter, c, impacts this optimization problem, defining the extent of the margins. A small c value leads to narrow margins, indicative of high fidelity to the data and low variance but high bias, with few support vectors. Conversely, a large c value results in wider margins, potentially accommodating more misclassifications, leading to lower bias, higher variance, and more support vectors.

For non-linear class boundaries, the feature space is expanded using higher-order polynomial functions of the predictors. SVM offers several advantages, including efficiency in high-dimensional spaces, utilization of only a small subset of training points (known as support vectors) in the decision function, and the ability to specify various kernel functions for the decision function.

b) Random Forest (RF) machine learning technique involves the aggregation of multiple decision trees. Aggregation is a method to mitigate high variance by combining the outcomes of several decision trees through a majority voting process in classification, thus enhancing prediction accuracy. The procedure includes creating multiple subtraining sets, constructing individual prediction models using these sets, and then averaging the prediction results through majority voting. However, slight variations in the data can lead to significant changes in the final estimated tree. Mathematically, if the results of the individual prediction models are denoted by:

$$C'(a), C''(a), C'''(a), \ldots C^k(a) \tag{3}$$

The average prediction $C_{avg}(a)$ can be computed as:

$$C_{avg}(a) = \frac{1}{k}\sum_{i=1}^{k} C^k(a) \tag{4}$$

Given the challenge of obtaining large datasets, the concept of bootstrapping aggregation, often referred to as bagging (repeated sampling with replacement), is commonly employed. The number of estimators is typically represented by the parameter k, where a value of one hundred (100) is often sufficient for achieving satisfactory performance. In random forest classification, a useful parameter for estimating test error is the Out-of-Bag (OoB) error. On average, each bagged tree utilizes two-thirds of the observations, leaving one-third as Out-of-Bag observations, which are not used for fitting. Predictions can be made for each ith observation in the Out-of-Bag set, followed by a majority vote for classification. The OoB approach for estimating test error is particularly advantageous when bagging large datasets, as traditional cross-validation may be challenging. Decision trees offer key advantages, including their simplicity in explanation and their resemblance to human decision-making processes.

c) Naïve Bayes (NB) classifier operates on three principles: the utilization of conditional probability, Bayes' theorem, and the assumption of feature independence. Probability denotes the likelihood of event occurrence, and Naïve Bayes employs the following conditional probability equation represented as:

$$P(D|E) = \frac{P(D,E)}{P(E)} \tag{5}$$

where $P(D,E)$ denotes the intersection between D and E, and $P(E)$ is the probability of E. $P(D|E)$ signifies the probability of D occurring given that E has already occurred. The relationship between $P(D|E)$ and $P(E|D)$ can be expressed via Bayes' theorem, articulated as:

$$P(E|D) = \frac{P(D|E).P(E)}{P(D)} \tag{6}$$

The Naïve Bayes classifier establishes a connection between input features and class based on probability. Given a set of features $X = \{X_1, X_2, X_3 \ldots, X_n\}$ and the objective is to forecast the class X, the approach is to identify the X that yields the highest $P(Y|X)$. Examining all features for each class can be intricate; thus, the optimal strategy is to adopt Bayes' theorem. For this scenario, Bayes' theorem can be expressed as:

$$P(Y|X) = \frac{P(X|Y).P(Y)}{P(X)} \tag{7}$$

Since $P(X)$remains constant across all classes as it is independent of Y, attention now focuses on determining the values for $P(X|Y)$ and $P(Y)$, which can be estimated from the data. $P(Y)$ derived from the training data is defined as:

$$P(Y) = \frac{\text{Number of Samples Labeled Y}}{\text{Total Number of Samples}} \tag{8}$$

For $P(X|Y)$, the independence assumption is employed, stating that individual features are independent. It is expressed as:

$$P(X_1, X_2, \ldots, X_n|Y) = P(X_1|Y).P(X_2|Y) \ldots P(X_n|Y) \tag{9}$$

This classifier offers several advantages: it is fast and uncomplicated to implement; it scales effectively, requiring minimal parameters, and the feature probabilities can be computed in parallel due to their independence.

B. Deep Learning Model

Recently, there has been a transition from conventional methods towards employing deep neural networks for image classification tasks. Often, the traditional classifiers encounter limitations when dealing with extensive datasets. The sheer volume of data being generated has been increasing exponentially in recent years. Moreover, with the emergence of faster processing units, CNNs, a class of deep learning models well-suited for image classification tasks have become the standard approach. Deep learning represents a subset of machine learning involving the creation of CNN models capable of learning intricate abstractions or representations of given data and leveraging this information for qualitative and quantitative predictions. Essentially, deep learning plays a crucial role, particularly in computer vision, as CNNs can autonomously extract patterns or features from training data.

Despite the proliferation of large datasets across various domains, image classification models typically contend with limited data, ranging from a few hundred to a few thousand samples. While training from scratch with this limited data is feasible, achieving optimal accuracy is often challenging. Deep learning endeavors to enhance outcomes while reducing execution time, computational complexity, and implementation costs. Numerous algorithms have been devised to mitigate costs, yet accuracy remains a paramount concern for image classification. Generally, neural network accuracy is influenced by factors such as the quantity of training samples, overfitting (wherein the network performs well on the provided data but struggles to generalize to unseen data), regularization techniques (such as data augmentation and dropout to alleviate overfitting), and model parameters (including the number of filters per convolution layer and the networks' depth). It is worth noting that neural networks are computationally demanding and require significant memory resources, often necessitating high-speed Graphics Processing Unit (GPUs) or Tensor Processing Units (TPUs) for efficient execution. This challenge has been partly addressed through the concept of transfer learning, wherein pre-trained models with learned parameters are made available for other classification tasks. This approach arose due to the extensive computational resources and time required to train these models initially, resources that are often inaccessible to most users. This helps to expedite model training by leveraging pre-existing knowledge and fine-tuning the network's parameters on our specific drowsiness detection task, thereby reducing the need for extensive training data.

Due to the limited size of the dataset, data augmentation techniques were employed to augment the dataset. Subsequently, the data was partitioned into three subsets for training, validation, and testing, respectively, following a percentage ratio: 70% for training, 15% for validation, and 15% for testing.

The preprocessing operation of the training data encompasses various tasks, including reading the image files, decoding the image content into RGB pixel grids, converting the pixels into floating-point tensors, and subsequently rescaling the pixel values to the [0,1] interval, as neural networks preferably operate on small input values. The Keras package provides utilities to facilitate these preprocessing steps. Specifically, the ImageDataGenerator utility sets up Python generators that automatically convert image files into batches of preprocessed tensors. Data augmentation is employed to generate additional samples from existing training data by applying diverse transformations. The objective is to expose the model to varied training data, thereby enhancing its robustness, and improving its ability to generalize to new data. Moreover, data augmentation helps mitigate the risk of overfitting, which arises when there are insufficient samples for learning. Several transformations were considered during data augmentation, resulting in images with channel shifts, zoom-ins of 0.2, height shifts of 0.2, rotations of 45°, width shifts of 0.2, horizontal flips, and shear angle changes to 45^o. Ultimately, the total number of images available amounted to 13,032. The training set comprised 4,554 drowsy and 4,567 non-drowsy images, while the validation set contained 976 drowsy and 979 non-drowsy images. The test set encompassed 977 drowsy and 979 non-drowsy images.

C. VGG16 architecture

The VGG16 architecture was developed by Simonyan and Zisserman [18]. This architecture is renowned for its ease of implementation and applicability to various image classification tasks, rendering it a widely adopted CN architecture. Figure 2 illustrates the structure of the VGG16 network, which can be divided into two components: the convolutional base, trained on ImageNet and comprising a series of pooling and convolutional layers, and the classifier base, contingent upon the number of classes. Typically, the network comprises 16 primary layers, including 13 convolutional layers and 3 dense layers. To alleviate the computational burden, a pre-trained model was employed-essentially a pre-saved network trained on a large dataset (ImageNet). Training a VGG16 model from scratch entails dealing with approximately 15 million parameters, which is computationally demanding and necessitates 2-4 GPUs over several days or weeks. Consequently, transfer learning was introduced, allowing for the utilization of pre-trained weights of different classification tasks without the need for retaining albeit requiring modifications to the classifier base to accommodate the required number of classes for a specific problem at hand.

The convolutional base is employed due to the broad and reusable patterns learned at this stage, encompassing generic features such as colors, visual edges, and textures. We transferred the learned parameters from the convolutional base onto our datasets. However, the patterns earned by the classifier base are more tailored to the specific classes in which the model was trained. Thus, we had to train our dense base on top to adapt to the binary classi-

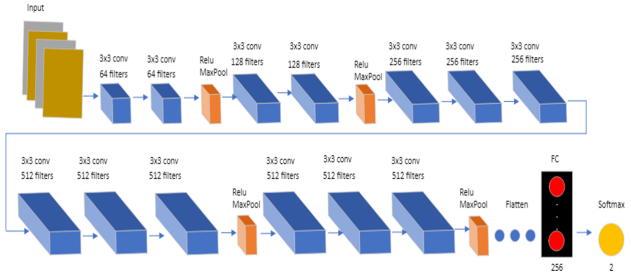

Figure 2. Schematic of the the VGG16 architecture.

fication task. Consequently, our modified CNN comprises 13 convolutional layers and 2 dense layers. The number of filters in convolution layers increases (multiples of 64) as we progress deeper into the network, ranging from 64 to 512 filters. Each convolutional layer in the network is a 3×3 grid with varying filters. The increase in filter count with depth is expected as the level of abstraction and the number of features to extract grow. Additionally, generality diminishes as the convolutional layers deepen. Rectified Linear Unit (ReLU) activation is applied with Maxpooling on the convolved outputs. While three main activations–sigmoid, hyperbolic tangent (tanh), and ReLU–are typically used for hidden layers, ReLU is preferred due to its avoidance of the vanishing gradient problem during backpropagation, ultimately yielding superior model accuracy. Following the last convolution, the output is flattened into a single stretch of neurons known as the dense layer. The first dense layer consists of 256 neurons, while the second dense layer contains 2 neurons, representing the binary classification output. In the output layer, we consider three activations: linear, sigmoid, and SoftMax. Although sigmoid or SoftMax can be employed for classification problems, SoftMax activation, which converts a vector of numbers into a vector of probabilities, was chosen in this study due to its superior accuracy.

The VGG16 model, integrated into Keras, operates with an input image size of 224×24 pixels. Fine-tuning was performed by adding a custom classifier base on top of the convolutional base, freezing the base network, training the added part, unfreezing some layers in the base network, and jointly training both these layers and the classifier. Weights are computed solely for the convolutional layers. Additionally, after each convolution operation, the output image assumes the depth of the convolved filter. Max-pooling layers do not possess parameters/weights; instead, they extract the maximum value representing a section of the output shape, thereby reducing the dimension of the output shape.

Parameter specifications are essential for training the classifier base to enhance prediction accuracy. These parameters include the cost function, optimizer type, and learning rate. The cost function penalizes deviations between predicted and expected labels, to minimize incorrect predictions. Various loss functions, such as mean squared error (MSE) and binary cross-entropy, can be utilized depending on the problem type. Optimizers adjust neural network weights to minimize cost, significantly impacting the training outcome. Adaptive optimizers, such as Adaptive Moment Estimation (Adam), offer advantages over traditional gradient descent algorithms by adjusting the learning rate dynamically. Table Ishows the parameter values used in training the classifier base.

TABLE I. Parameters for training classifier base

Parameter	Choice
Loss Function	Binary crossentropy
Optimizer	Adam
Learning Rate	$2e-6$

D. Performance Evaluation

After the entire models are trained, the final step as seen in Figure 1 is to evaluate the performance. Now, the test data is passed through the models. Again, different performance metrics have been developed to analyze results. For the traditional classifiers, the metrics include accuracy, recall, precision, and F-score. These metrics provide quantitative measures of the model's effectiveness in distinguishing between drowsy and non-drowsy states.

Accuracy stands out as a prominent performance metric in machine learning, particularly in scenarios with unbiased class distribution. It gauges the classifier's capability to assess, scrutinize, and discern relationships, patterns, and variations among the features defining a dataset. The accuracy measure heavily relies on the input data and the classifier's adeptness in leveraging learned features to enhance predictions for unseen data. Mathematically, accuracy is represented as shown in the following equation:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

Where TP is true positive, TN is true negative, FP is false positive, and FN is false negative. TP refers to the number of images with an expected drowsy label and correctly predicted with a drowsy label. TN also refers to the number of images with an expected non-drowsy label and correctly predicted with a non-drowsy label. However, FP refers to the number of images assigned a drowsy label rather than the correct expected non-drowsy label. Also, FN refers to the number of images assigned a non-drowsy label rather than the correct expected drowsy label.

In machine learning, recall measures how correctly the model predicted or found correct positive responses (i.e., TP) against the total number of expected correct responses. It is mathematically expressed by Eq. (11) as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

Precision measures how the model found correct pos-

itive responses (TP) to the total number of positive responses. It is mathematically expressed by Eq. (12) as

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

F–score is the weighted average of precision and recall. For even class distribution, accuracy is an ideal performance measure, while an F-score is the best measure of a system's performance for uneven class distribution. F-score is also another measure of a test's accuracy given mathematically as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+ Recall}} \tag{13}$$

According to [19], accuracy and sensitivity are the main measures. Sensitivity describes cases where drowsiness is present, and this is a significant consideration as the driver ought to be notified when in a drowsy state. However, to evaluate performance, there is the need to obtain the confusion matrix; this is a square matrix diagram with information on the various correct and incorrect classifications made by the classifiers. The performance of the neural networks is evaluated based on the loss and accuracy.

In terms of cost and execution time, traditional machine learning methods such as SVM, NB, and Random Forest can typically be executed on the CPU, which is generally more cost-effective and readily available compared to GPU resources. However, for deep learning models like VGG16, large-scale training tasks often necessitate GPU resources due to the computational intensity involved. While running traditional models may be more feasible in terms of resource requirements, the VGG16 architecture, with its numerous layers and complex structure, demands more time for training compared to traditional models. Despite the longer training duration, leveraging a sufficiently powerful system for VGG16 training can enhance its reliability and accuracy in drowsiness detection compared to traditional models. This is attributed to the deeper understanding and feature extraction capabilities of deep learning architectures like VGG16, which can lead to more robust and nuanced detection outcomes. Therefore, while traditional methods may offer advantages in terms of resource efficiency, the potential performance gains achieved by VGG16 justify the investment in computational resources for training.

3. Results and Discussion

A learning curve exhibits an estimator's validation and training scores across different counts of training samples. It serves as a tool to assess the potential benefits of additional training data and to gauge whether the estimator is prone to bias or variance errors. Each estimator has benefits and disadvantages. Bias and variance can be used to break down the generalization error. An estimator's bias is represented by its average error across many training sets. An estimator's variance reveals how responsive it is to various training sets. Ideally, a dataset is grouped into three: training dataset, validation dataset, and testing dataset. Training takes place on the training set, followed by evaluation on the validation set. When it appears that the experiment has been successful, a final evaluation of the test set may be conducted. Nevertheless, splitting the available data into three sets significantly reduces the number of samples available for model training. The outcomes may fluctuate based on the randomization of the (train, validation) set pairs.

Cross-validation (CV) is a technique that can be used to solve this issue. When doing a CV, the validation set is no longer required, but a test set should still be kept aside for final assessment. Therefore, the training set is divided into k smaller sets in the fundamental strategy, known as a k-fold CV. For each of the k "folds," the procedure is as follows as seen in Figure 3:

- A model is trained using k-1 of the folds.
- The resulting model is validated on the remaining fold (i.e., used as a test set to compute performance).

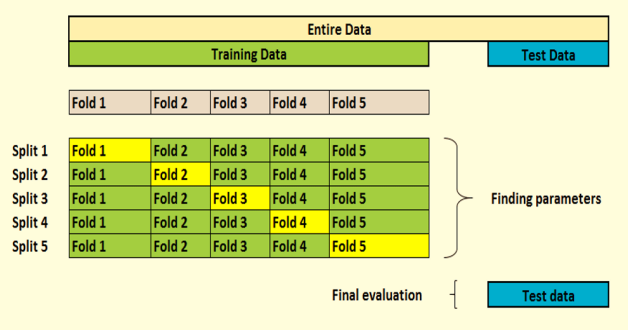

Figure 3. Illustration of the K-fold cross-validation.

The performance indicator provided by k-fold cross-validation is derived from the average of results obtained during the loop iterations. Despite its potential computational expense, this method efficiently utilizes the available data without excessive wastage (unlike the fixed random validation set), which proves advantageous especially when dealing with limited samples.

Figure 4 shows the learning curves for the tested classifiers. For the NB classifier, the training score declined as the number of samples increased while the cross-validation score was approximately constant. With increasing training set size, the validation score and training score for the naïve Bayes algorithm converge to a pretty low number. Therefore, adding more training data is probably not going to help much. For the SVM classifier, the training score was constant, with an accuracy score of 100, while the CV score increased with an increase in the number of training examples. In other words, the SVM's training score is significantly higher than its validation score for small amounts of data. Increased generalization will probably result from adding additional training data. For the RF classifier, the training score stayed constant as well, with an accuracy

score of 100, while the cross-validation score increased with an increase in the number of training examples. This means that with small amounts of data, the RF's training score is substantially higher than its validation score. More training samples will almost certainly result in more generalization.

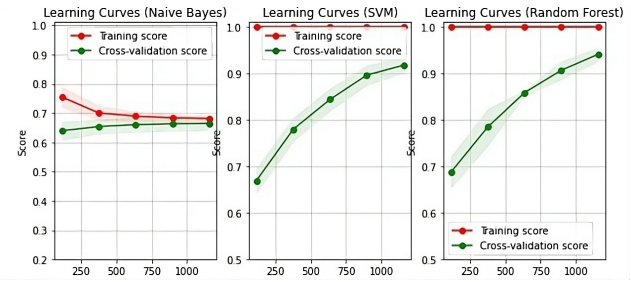

Figure 4. Learning curves for the tested classifiers

Scalability describes how the model can learn, that is, the rate it can fit the model to a training data size. Often the goal is to have a model that can learn fast without consuming much memory, irrespective of the training data size. Figure 5 shows the scalability of the classifiers, a plot of the training examples against the fit_times. Naïve Bayes took less time fitting the model to different training sample sizes (0 - 80 milliseconds), followed by SVM (0 – 2.4 seconds). On the other hand, it took a time from (25 milliseconds – 2 seconds) for RF to fit all the training samples into the model.

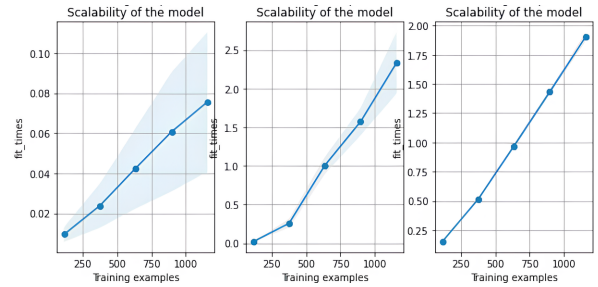

Figure 5. Scalability of the tested classifiers.

Figure 6 shows the performance of the three models, a plot of the test score against the "fit_times". For NB, the accuracy score increases as the fit time increases. For SVM, the accuracy score increases as fit_times increases. Similarly, the accuracy score for RF increases as the fit time increases.

Figures 7– 9 are a graphical view of the confusion matrix for SVM, Random Forest, and Naïve Bayes. In the algorithm, the drowsy class was represented as binary zero, while the non-drowsy class was represented as binary one. The binary representation on the left side represents the predicted label and that at the bottom represents the actual label. A total of 435 samples were used for testing

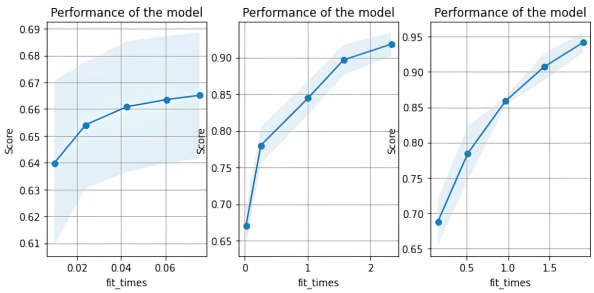

Figure 6. Performance of all three classifiers.

each classifier. The positive sample represents images with drowsy labels, while the negative represents images with non-drowsy labels.

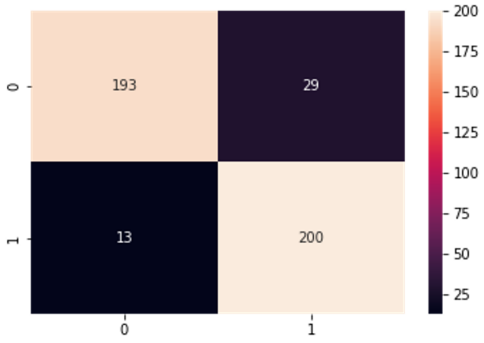

Figure 7. Confusion matrix for SVM.

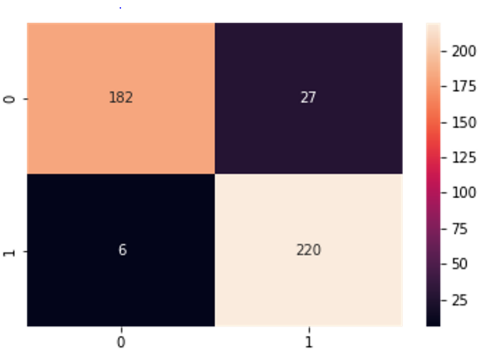

Figure 8. Confusion matrix for random Forest.

For the SVM classifier, 193 images were predicted as True Positives, 200 as True Negatives, 29 as False Positives, and 13 as False Negatives. Using the Random Forest classifier, True Positives were 182, 220 predicted True Negatives; False positives and Negatives were 27 and 6, respectively. Next, the Naïve Bayes classifier had the number of True Positives, True Negatives, False Positives, and False Negatives to be 93, 209, 116, and 17, respectively.

Table II shows the performance after testing the three classifiers and from the table, Random Forest recorded the highest accuracy (92.41), followed by SVM (90.34) and

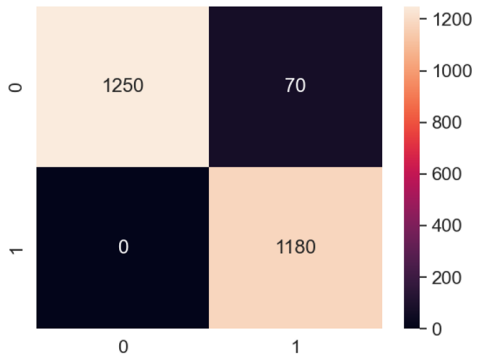

Figure 9. Confusion matrix for Naïve Bayes.

Naïve Bayes classifier (69.43). In this research, accuracy is a good measure of performance because an even class distribution was used. However, the F-score, a similar and relevant performance metric, was considered for additional justification of results. The results also show RF having the highest F1-score (93.02) and the lowest (75.86) by Naïve Bayes. NB had the lowest accuracy, and a plausible reason is that the independence assumption may not always hold as there are no model interactions between the features. As a result, it can limit classification power. it is seen that Naïve Bayes has the lowest precision (64.31), while SVM and RF have the closest precision. This means that given 100 test images, SVM correctly predicted approximately 89 and RF approximately 87 with drowsy labels.

TABLE II. Accuracy of the classical machine learning algorithms. RF: random forest;SVM: support vector machine; and NB: Naive Bayes.

Classifier	Accuracy	Precision	Recall	F1-Score
RF	92.41	89.07	97.35	93.02
SVM	90.34	87.34	93.89	90.50
NB	69.43	64.31	92.48	75.86

Next, we look at the neural network performance shown in Figure 10. The figure is the plot of accuracy against the number of epochs with a standard batch size of 32. It is observed that the validation accuracy plot closely follows the training accuracy, which shows that the network is well-trained and can generalize well. The training accuracy rises from 0.52 to a final accuracy of 1.0, while the training loss reduces from 0.6319 to 0.0024, respectively.

Figure 11 plots the training and validation loss against the number of epochs. Generally, training and validation loss is expected to decline for a good network model as the network learns from the data. The training and validation loss is close, which is evidence of a good, trained network.

After training and validating the neural network, the network is tested with the test data. The confusion matrix for the neural network is shown in Figure 12. The left binary representation is the predicted label whereas that

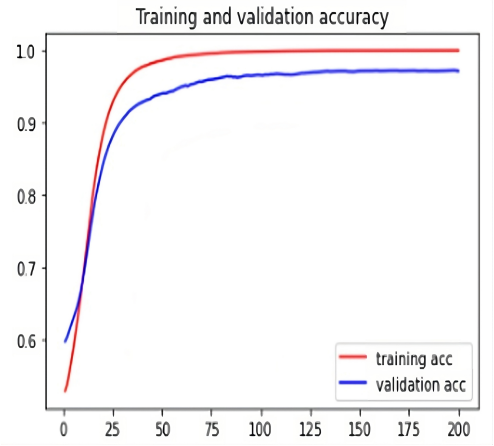

Figure 10. Plot for training and validation accuracy (200 epochs).

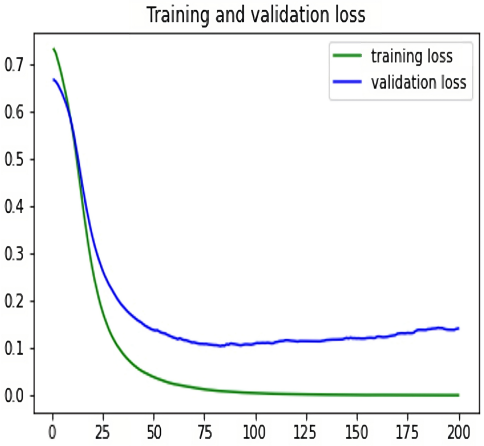

Figure 11. Plot for training and validation loss (200 epochs).

at the bottom represents the actual labels. Binary zero (0) represents a drowsy label while binary one (1) represents non-drowsy labels. The network correctly predicted the actual labels for drowsy images (1250) and incorrectly predicted (70) of the non-drowsy images as drowsy. A total of 1180 non-drowsy images were predicted correctly. In summary, a total of 2500 images were tested on the network and the confusion matrix is also shown in Figure 12

The accuracy, precision, recall, and F-score are computed for this network and summarized in Table III. The accuracy achieved for the test was 97.20. The model correctly predicted all drowsy cases, which is very important in detection. Deep neural networks have outperformed the three traditional machine learning methods, making it the most preferred model for image classification.

TABLE III. Performance of VGG16 network

Classifier	Accuracy	Precision	Recall	F1-Score
VGG16	97.20	100	94.70	97.28

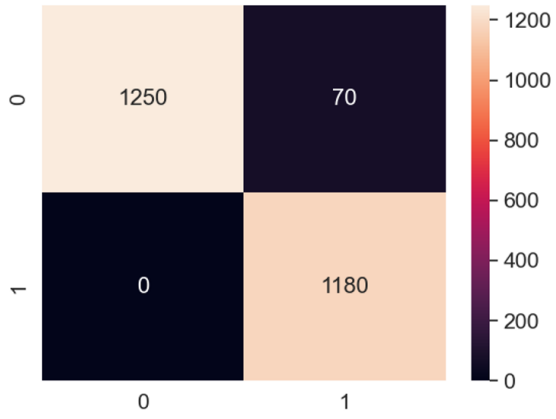

Figure 12. Confusion matrix for the neural network.

4. Conclusion and Future work

This research investigates drowsiness detection using three conventional machine learning classifiers and a deep neural network. While prior research relied on recorded EAR and MOR values from images as classifier input, this study explores the utilization of image pixels, yielding notable accuracy. Unlike in previous literature, which relies on the accuracy of recorded values, the novel approach of utilizing image pixels derives input values directly from facial geometry, yielding promising results. Additionally, the deep neural network demonstrates exceptional accuracy in drowsiness classification, surpassing all traditional methods with an accuracy of 97%. Notably, all approaches are non-invasive and cost-effective, making them suitable for integration into automobile dashboards for convenient drowsiness detection. While this research has made significant strides in drowsiness detection using machine learning classifiers and deep neural networks, it is essential to acknowledge certain limitations and identify potential avenues for future research. The study may have been constrained by the size of the dataset used for model training and testing. Expanding the dataset size, possibly collecting more diverse samples from a wider range of demographic groups and environmental conditions, could enhance the model's robustness and generalization capabilities. Although utilizing image pixels directly from facial geometry yielded promising results, the input features may still lack certain nuances captured by more complex physiological measurements. Future research could explore the integration of additional biometric signals or multimodal data sources to further improve detection accuracy. While accuracy is a commonly used metric for assessing model performance, it may not provide a comprehensive understanding of the system's effectiveness in real-world scenarios. Future studies could consider incorporating additional metrics such as receiver operating characteristic (ROC) curves to evaluate the model's performance across different thresholds and imbalanced datasets. Integrating the developed drowsiness detection system into automobile dashboards or other real-time applications may pose implementation challenges related to hardware compatibility, latency, and user interface design. Future research should address these practical considerations to ensure seamless deployment and user acceptance. As deep neural networks often require significant computational resources for training and inference, optimizing the model architecture and training process to improve computational efficiency is crucial, especially for deployment in resource-constrained environments. Future work could explore techniques such as model compression, quantization, and hardware acceleration to mitigate computational costs. In conclusion, while this study lays a strong foundation for drowsiness detection using machine learning and deep learning approaches, there remains ample room for further research to address the limitations and enhance the practicality, accuracy, and scalability of the developed system

Acknowledgment

The authors acknowledge the support of the Louisiana Transportation Research Center in supporting the work performed in this paper through LTRC Project #22-2TIRE and the Center for Equitable Artificial Intelligence and Machine Learning Systems (CEAMLS) at Morgan State University under Project #11202202. The authors would also like to thank the support of Southern University and A&M College for the tremendous support provided to finalize this work.

References

[1] E. Essel, F. Lacy, W. Elmedany, F. Albalooshi, and Y. Ismail, "Driver drowsiness detection using fixed and dynamic thresholding," in *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*. IEEE, 2022, pp. 552–557.

[2] 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. 61 904–61 919, 2019.

[3] S. Mehta, P. Mishra, A. J. Bhatt, and P. Agarwal, "Ad3s: Advanced driver drowsiness detection system using machine learning," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*. IEEE, 2019, pp. 108–113.

[4] W. Kong, L. Zhou, Y. Wang, J. Zhang, J. Liu, S. Gao *et al.*, "A system of driving fatigue detection based on machine vision and its application on smart device," *Journal of Sensors*, vol. 2015, 2015.

[5] Y. Chellappa, N. N. Joshi, and V. Bharadwaj, "Driver fatigue detection system," in *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*. IEEE, 2016, pp. 655–660.

[6] M. Awais, N. Badruddin, and M. Drieberg, "A hybrid approach to detect driver drowsiness utilizing physiological signals to improve system performance and wearability," *Sensors*, vol. 17, no. 9, p. 1991, 2017.

[7] 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)*. IEEE, 2020, pp. 237–242.

[8] R. Manoharan and S. Chandrakala, "Android opencv based effective driver fatigue and distraction monitoring system," in *2015 International Conference on Computing and Communications Technologies (ICCCT)*. IEEE, 2015, pp. 262–266.

[9] W. Deng and R. Wu, "Real-time driver-drowsiness detection system using facial features," *Ieee Access*, vol. 7, pp. 118 727–118 738, 2019.

[10] N. Kuamr, N. C. Barwar, N. Kuamr *et al.*, "Analysis of real time driver fatigue detection based on eye and yawning," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 6, pp. 7821–7826, 2014.

[11] S. Viswapriya, S. Balabalaji, and Y. Sireesha, "A machine-learning approach for driver-drowsiness detection based on eye-state," *International Journal Of Engineering Research & Technology (IJERT)*, vol. 10, no. 04, 2021.

[12] I. Latreche, S. Slatnia, and O. Kazar, "Cnn-lstm to identify the most informative eeg-based driver drowsiness detection brain region," in *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2022, pp. 725–730.

[13] H. Lamaazi, A. Alqassab, R. A. Fadul, and R. Mizouni, "Smart edge-based driver drowsiness detection in mobile crowdsourcing," *IEEE Access*, vol. 11, pp. 21 863–21 872, 2023.

[14] Y. Jebraeily, Y. Sharafi, and M. Teshnehlab, "Driver drowsiness detection based on convolutional neural network architecture optimization using genetic algorithm," *IEEE Access*, 2024.

[15] A. S. Agarkar, R. Gandhiraj, and M. K. Panda, "Driver drowsiness detection and warning using facial features and hand gestures," in *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*. IEEE, 2023, pp. 1–6.

[16] R. K. Peddarapu, B. Likhita, D. Monika, S. P. Paruchuru, and S. L. Kompella, "Raspberry pi-based driver drowsiness detection," in *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, vol. 5. IEEE, 2024, pp. 864–869.

[17] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, "Yawdd: A yawning detection dataset," in *Proceedings of the 5th ACM multimedia systems conference*, 2014, pp. 24–28.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] I. Gupta, N. Garg, A. Aggarwal, N. Nepalia, and B. Verma, "Real-time driver's drowsiness monitoring based on dynamically varying threshold," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE, 2018, pp. 1–6.

Ebenezer Essel, M.Sc., received the B.Sc. degree in electrical engineering from Kwame Nkrumah University of Science and Technology, Ghana, in 2019; and the M.Sc. degree in electrical engineering from Southern University and A&M College, Baton Rouge, USA, in 2022. He is currently pursuing his Ph.D. degree in electrical engineering and has been a Graduate Research Assistant with the Radio Frequency Communications Laboratory, at Louisiana State University, Baton Rouge, USA, since 2023. His research interests include machine learning, intelligent systems, analog electronics, and radio frequency communication.

Abeer AbdElhamid, M.Sc., received her M.Sc. degree in Electronics and communication Engineering, ECE department, Faculty of Engineering, Mansoura University, Egypt, in 2022. She is currently pursuing her Ph.D. studies with the ECE department, Faculty of Engineering, Mansoura University. Abeer has more than 4 years of hands on experience in the field of image/signal processing and analysis, focused on the application of AI/ML for medical data analysis for disease diagnosis.

Mahmoud Darwich, Ph.D., is a distinguished academic and researcher in the field of Computer Science, currently serving as an Assistant Professor at the University of Mount Union, Ohio, USA. He received his Bachelor's degree from Beirut Arab University, Lebanon, in 2006. Subsequently, he completed his master's and doctorate degrees in Computer Engineering from the University of Louisiana at Lafayette in 2013 and 2017, respectively. His scholarly portfolio includes over 30 publications in prestigious book chapters, journals, and conferences, reflecting his extensive research in the realm of video streaming technologies. Before his tenure at the University of Mount Union, he was a faculty at Bloomsburg University of Pennsylvania and Navajo Technical University in New Mexico.

Fahmi Khalifa, Ph.D., received his B.Sc. and M.Sc. degrees in Electronics and Electrical Communication Engineering from Mansoura University, Egypt in 2003 and 2007, respectively. He received his Ph.D. degree in 2014 in Electrical Engineering from ECE Department, University of Louisville, USA. Currently, he is an Assistant Professor with Electrical and Computer Engineering,Morgan State University, USA. He has more than 15 years of hands-on experience in the fields of image processing, machine learning, medical image analysis, computer-aided diagnosis, and digital and analog signal processing. Dr. Khalifa has authored/co-authored more than 190 peer-reviewed publications appearing in high-impact journals, selective peer-reviewed top-rank international conferences, and leading edited books.

Fred Lacy, Ph.D., is a professor and chair of the Electrical Engineering Department at Southern University. He earned his Ph.D. in Electrical Engineering from Howard University in 1993, followed by a postdoctoral fellowship at the University of California, San Diego. Prior to his current position, he worked as a medical device reviewer at the FDA. Dr. Lacy's research spans multidisciplinary projects in electrical engineering, biology, chemistry, agriculture, physics, and computer science, with a focus on developing electronics-based microsensors. He has collaborated with federal agencies like the DoD and DoE on national security-related research, involving remote sensing, signal processing, and machine learning for threat detection. Additionally, he is dedicated to enhancing engineering education and opportunities for underrepresented minorities in STEM fields.

Yasser Ismail, Ph.D., serves as an Associate Professor in the Electrical Engineering Department at Southern University and A&M College (SU), with over two decades of experience in teaching and pioneering research. He holds master's and doctoral degrees in Computer Engineering from the University of Louisiana at Lafayette. Dr. Ismail's expertise encompasses machine learning-based applications, hardware accelerators, VLSI and FPGA systems design, Cloud Computing, Cybersecurity, IoVT, digital video processing, and wireless communication. With a portfolio of over 100 publications, he has made significant contributions to these fields. Dr. Ismail's involvement in national security projects, including collaborations with state and federal agencies, highlights his commitment to advancing critical technologies. He is also dedicated to mentorship, having guided numerous students in state and federal projects, fostering their academic and professional growth. Dr. Ismail plays a key role in shaping Electrical and Computer Engineering education, developing innovative courses, and contributing to curriculum design. Beyond academia, he is actively engaged in STEM initiatives, organizing programs for both university and K-12 students to inspire the next generation of innovators.