IJCDS 1571010177

International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

High-Fidelity Machine Learning Techniques for Driver Drowsiness Detection

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

[1] Department of Electrical & Computer Engineering, Louisiana State University Baton Rouge, LA, U.S.A.
[2] Department of Electrical and Computer Engineering, Morgan State University, Baltimore, MD, U.S.A.
[3] Department of Mathematics and Computer Science, University of Mount Union, Alliance, Ohio, U.S.A.
[4] Department of Electrical Engineering, Southern University and A&M College, Baton Rouge, LA, U.S.A.

E-mail address: eessel1@lsu.edu, beroabdo345@gmail.com, darwicma@mountunion.edu, fahmi.khalifa@morgan.edu, fred_lacy@subr.edu, yasser_ismail@subr.edu

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

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. 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][5], 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. [6], 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

E-mail: yasser_ismail@subr.edu

classifier were used together with the devices for detection, with an achieved detection accuracy of 80.55%.

Awais el al. considered biological parameters such as the heart rate, time-domain, and frequency domain measures extracted from electroencephalogram (EEG) and electrocardiogram (ECG) combined with the SVM and k-strongest strengths (kSS) to detect drowsiness [7]. 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 [8], used the deep learning method and achieved an accuracy of 83.30%. Essel's study in [1] draws the method and main contributions from [9] and the model's applicability to Android devices. Deng and Wu [10] 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. [11] 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%.

Reddy et al. [12] researched a deep CNN for driver drowsiness detection based on eye state. A dataset of 1200 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%. Reddy et al. [12] 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.

This paper is an elaborate study on the use of machine learning techniques for driver drowsiness detection. The focus is on comparing various ML and deep learning for high-fidelity detection. 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 approaches to implement a high-fidelity drowsiness detection algorithm. A schematic of the pipeline is shown in Fig. 1. The subsequent sections provide comprehensive insights into the utilization of these methodologies.

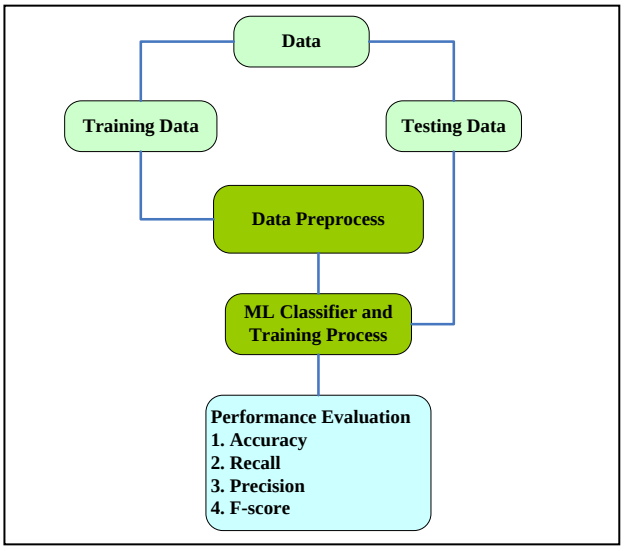

Figure 1. Schematic of the proposed pipeline flow and phases.

A. Machine Learning Methods

This research presents a detection model that leverages ML classifiers for enhanced efficiency. The proposed model along with its implementation phases: (1) data acquisition, (2) data preprocessing, (3) ML classifier and learning operation, and (4) performance evaluation, are seen in Figure 1. Each phase plays a crucial role in the overall functionality of the system, ensuring robust detection capabilities and rigorous performance assessment.

1) Data Acquisition: As seen in Figure 1, the model starts with data acquisition, and ML algorithms are designed to work on static frames; the data was obtained from [13], which consists of participants from different ethnicities and genders. The data consists of individuals whose images can be categorized under different conditions, including persons with and without glasses, yawning, and no yawning. Next, the data was split into two, one for training and the other for testing.

2) Data Preprocessing: In this phase, the training data had to be processed before passing to the classifier, and this was done by resizing the images to 32 x 32 to reduce processing time and use less memory. The associated label for each image was also converted into a binary value. The drowsy state was labeled as a binary

"zero", whereas the non-drowsy state was labeled as a binary "one".

3) Machine Learning Classifier and learning operation: After the training data has been processed, it is ready to be used for training the classifiers. The ML classifiers considered are that of the supervised form of learning. The three classifiers used are SVM, NB, and RF and their details are given below.

a) Support Vector Machine (SVM) is one of the early ML classification methods dating to the early 1990s and is a generalized form of the 'maximal margin classifier'. Support Vector Machine was initially limited to cases with linear boundaries. However, improvement has seen its application to a wide range of datasets. SVM uses the concept of a hyperplane, and in n-dimensional space, a hyperplane is a flat affine subspace of dimension (n-1). In two-dimensional (2D), the hyperplane is a line, whereas in three-dimensional (3D), the hyperplane is a plane. In 2D, a hyperplane is defined by (1):

$$ax_1 + bx_2 + c = 0 \; or \; ax + b = 0 \qquad (1)$$

If the position of the training point lies on this equation, then the point is directly on the hyperplane. However, often, the training observations lie on either side of the hyperplane, which satisfies (2):

$$ax_1 + bx_2 + c < 0 \; or \; ax_1 + bx_2 + c > 0 \qquad (2)$$

The aim is to construct a hyperplane that separates the training sets perfectly according to their class labels. Then test data is classified based on the side of the plane where it lies. For example, suppose the data can be perfectly separated using a hyperplane, then an infinite number of such hyperplanes will exist. Thus, to construct a classifier based upon a separating hyperplane, there must be a reasonable way to decide from the infinite possible separating hyperplanes. Therefore, a hyperplane, which gives the maximum margin away from the training points, is chosen, termed the optimal separating hyperplane, and is done by computing the (perpendicular) distance from each training point to a given separating hyperplane. The smallest distance is the minimal distance of a training point away from the hyperplane, known as the margin. We can then classify a test observation based on which side of the maximal margin hyperplane it lies. Support vectors are training points closest to the hyperplane. These support vectors determine how far the margin can be set. In other words, they regulate the magnitude of the margin away from the hyperplane. In cases where we do not have a separating hyperplane, the concept of soft margin is introduced. This concept deals with allowing some of the training data to be on the wrong side just so the classifier will be good at classifying other observations. This tradeoff makes the classifier robust to different observations and better classifies the training data. A regularization parameter, C, a nonnegative tuning parameter, affects this optimization problem; it defines the limit of the margins. When C is small, narrow margins are developed, indicating the classifier is highly fitted to the data. Thus, having a low bias and a high variance with few support vectors.

On the other hand, when C is large, the margin is wider and subject to more wrong positioning. As a result, there is less hard fitting, high bias, and low variance with many support vectors. In non-linear class boundaries, the feature space is enlarged using higher-order polynomial functions of the predictors. Support vector machines have the following benefits: they are efficient in high-dimensional spaces, only employ a small portion of training points (called support vectors) in the decision function and can specify various kernel functions for the decision function.

b) Random Forest (RF) ML method stems from using several (aggregation) decision trees. Aggregation is a procedure to reduce high variance by totaling the result of several decision trees by majority vote in classification, thereby increasing the prediction accuracy. The process involves: making several sub-training sets, building separate prediction models using the training sets, and then averaging the prediction results by majority vote. However, a slight change in the data can cause a significant change in the final estimated tree. Mathematically, if the results for the individual prediction models are given by (3):

$$\alpha + \beta = \chi. \qquad (1) \qquad (1)$$

$$f^1(x), f^2(x), f^3(x), \ldots, f^n(x) \qquad (3)$$

The average of these will be given by (4):

$$f_{avg}(x) = \frac{1}{n}\sum_{n=1}^{n} f^n(x) \qquad (4)$$

Due to the difficulty of obtaining large datasets, the concept of bootstrapping aggregation (repeated sampling with replacement), popularly called bagging, is utilized. Often the number of estimators can be represented by the parameter, n. An n value of one hundred (100) is sufficient to achieve good performance. In random forest classification, a good parameter to estimate the test error is the Out-of-Bag (OoB) error. It is proven that, on average, each bagged tree utilizes two-thirds of the observations. The remaining one-third is not used for fitting and is termed the Out-of-Bag observations. Predictions can be made for each i^{th} observation in OoB observations, and then the majority vote is taken for classification. Also, the OoB approach for estimating the test error is particularly convenient when bagging on large datasets for which cross-validation is tough. The key advantages of decision trees include the ease of explanation and how they closely mirror human decision-making.

*c) Naïve Bayes (NB)*classifier utilizes three principles, that is, the use of conditional probability, Bayes' theorem, and feature independence assumption. Probability is the likelihood of event occurrence and Naïve Bayes uses the conditional probability equation given by (5) as:

$$P(A|B) = \frac{P(A,B)}{P(B)} \quad (5)$$

where $P(A,B)$ is the intersection between A and B, and P (B) is the probability of B. $P(A|B)$is referred to as the probability of A occurring given that B has already occurred. The relation between P (A|B) and P (B|A) can be represented through Bayes' theorem, which is stated by (6) as:

$$P(B|A) = \frac{P(A|B)*P(B)}{P(A)} \quad (6)$$

Naïve Bayes classifier relates input features to class based on probability. Given a set of features Y = {Y_1, Y2, Y3…, Yn} and the goal is to predict the class C, then look for C that gives the highest P (C|Y). It is difficult and complex to run through all features for each class; thus, the ideal approach is to resort to the following principle: Bayes' theorem. It can be defined for this case as:

$$P(C|Y) = \frac{P(Y|C) * P(C)}{P(Y)} \quad (7)$$

P(Y) is the same for all classes as it does not depend on C; it is a constant. Thus, now the focus will be to determine the values for P (Y|C) and P(C), which can be estimated from the data. P(C) obtained from the training data is defined as:

$$P(C) = \frac{Number\ of\ samples\ labeled\ C}{Total\ Number\ of\ samples} \quad (8)$$

For P (Y|C), we use the independence assumption that describes how the individual features are independent. It is given thus as:

$$P(X_1, X_2, \dots, X_n|C) = P(X_1|C) * P(X_2|C) * \dots * P(X_n|C) \quad (9)$$

The advantages of this classifier include: it is fast and straightforward to implement; it scales well; that is, it does not need several parameters, and the feature probability can be calculated in parallel since they are independent.

4) Performance evaluation: In assessing the performance of machine learning models, several key metrics are commonly employed, including accuracy, recall, precision, and F-score, as seen in Figure 1. Accuracy measures the overall correctness of the model's predictions, representing the proportion of correctly classified instances. Recall, also known as sensitivity, quantifies the model's ability to correctly identify all relevant instances within a dataset. Precision, on the other hand, assesses the model's accuracy in correctly identifying relevant instances among all instances classified as positive. The F-score, which combines precision and recall into a single metric, provides a balanced assessment of the model's performance, particularly useful when dealing with imbalanced datasets. Together, these metrics offer valuable insights into the effectiveness and reliability of machine learning algorithms in various applications. All these parameters will be elaborated in detail in the simulation results section.

B. Deep Learning Model

Recent years have seen a shift from the traditional method to applying a deep neural network for image classification. Often, the classifiers mentioned above become limited when you have a large data set. The volume of data being produced recently is growing exponentially. Also, with the advent of faster processing units, CNN has become the norm of the day. Deep learning is a machine learning form involving designing CNN models capable of learning diverse, intricate abstractions or representations of a given data and using that information to make qualitative and quantitative predictions. In other words, deep learning is pivotal and often employed in computer vision since convolutional neural networks can independently generate patterns/ features in the training data. Although large datasets are being produced in different fields, often, there is little data available for image classification models typically having data ranging from a few hundred to a few thousand. While it is possible to use this limited data to train content from scratch, often, one cannot achieve optimum accuracy. Deep learning aims to optimize results and reduce execution time, computational complexity, and implementation cost. Several algorithms have been developed to reduce cost, but accuracy is a key consideration for image classification. Generally, the accuracy of a neural network is affected by the following: (1) the number of training samples, (2) overfitting, that is, the network has specialized well in learning the given data but does poorly in generalizing to other unseen data (3) regularization (methods used to minimize the loss function to improve accuracy) – data augmentation and dropout are regularization methods used to correct overfitting, and (4) model parameters (that is, the number of filters per convolution layer, and the number of layers in the network). It is important to know that neural networks are computationally expensive and utilize ample memory space; thus, most networks need a high-speed graphics processing unit (GPU) or tensor processing unit (TPU) for execution. This problem has been addressed with the introduction of transfer learning, where models that have their parameters already trained are made available for other classification problems. This was done

because some of these models were trained using multi-parallel high-performance processors over several days, weeks, or months and these constraints are not available to most users. Figure 2 represents the steps of training the neural network model and evaluating its performance.

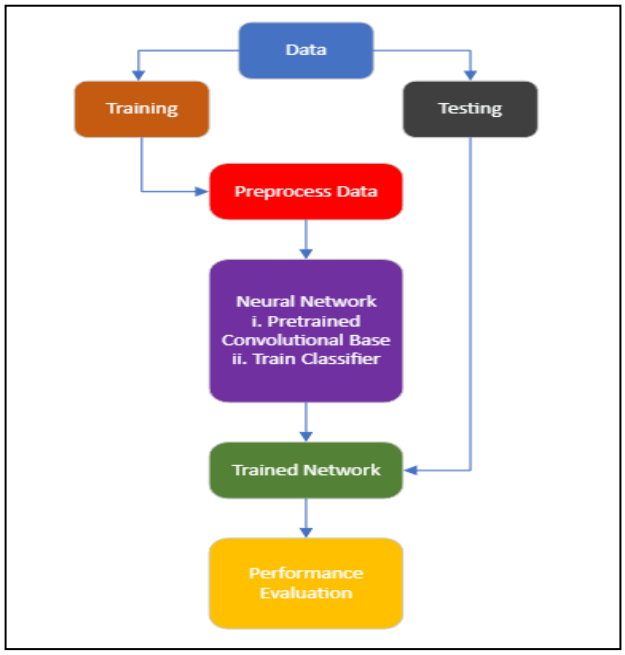

Figure 2. General Flowchart for Classification.

A. Data

The first step as can be seen in Figure 2 is the collection of data. Images were taken from the 'Yawning Detection Dataset' [13]. These individuals were from different ethnicities, including Caucasian, Asian, African, and Middle Eastern. The data was very small, so there was the need to use data augmentation to generate more data. After that, the data was divided into three for training, another for validation, and the last for testing in the percent ratio; 70% for training, 15% for validation, and 15% for testing.

B. Data Preprocessing

The next step in Figure 2 is to process the data before training the network. Data preprocessing involves reading the image files, decoding the image content to RGB grids of pixels, converting the pixels into floating-point tensors, and finally rescaling the pixel values (between 0 and 255) to [0,1] interval because, preferably, neural networks use small input values. Keras package offers utilities to handle these steps of preprocessing. The package has the ImageDataGenerator utility, which sets up Python generators that automatically turn image files into batches of preprocessed tensors. Data augmentation generates more samples from existing training samples by augmenting the data through different transformations. The aim is to expose the model to varying training data, thus making it robust and likely to generalize well on new data. This is also done to avoid the issue of overfitting which occurs when we have too few samples to learn from. Several transformations were considered, and some images produced were under the following: (1) channel shift, (2) zoom-in of 0.2, (3) a height shift of 0.2, (4) a rotation of 45°, (5) width shift of 0.2, (6) a horizontal flip, and (7) changing the shear angle to 45°. In the end, the total images available totaled 13,032. The training set had 4554 drowsy and 4567 non-drowsy images; the validation set had 976 drowsy and 979 non-drowsy images; the test set had 977 drowsy and 979 non-drowsy images.

C. Neural Network

Following the second block is the neural network and our study used VGG16 architecture developed by Karen Simonyan and Andrew Zisserman [15]. The network is easy to implement and applicable to many image classification problems, making it a widely used CNN architecture. Figure 3 shows the architecture of the VGG16 network. The network can be partitioned in two: (1) convolutional base trained on ImageNet – a series of pooling and convolutional layers, and (2) classifier base – depending on the number of classes. Generally, the network consists of 16 main layers (13 convolutional layers and 3 dense layers). A pre-trained model was used, a saved network previously trained on a large dataset (ImageNet) because the total weights to be trained for a VGG16 model is about 15 million parameters (this is computationally expensive). In addition, it requires about 2-4 GPUs to train over several days or weeks. Thus, transfer learning was introduced where it is possible to use the trained weights on different classification problems without having to retrain again but only to modify the classifier base to suit the number of classes required for the problem at hand.

The convolutional base is used since the representation or patterns learned in that stage were largely generic (colors, visual edges, and textures) and more reusable. We transferred the learned parameters from the convolution base onto our datasets. However, the patterns learned by the classifier base are more specific to the set of classes in which the model was trained, so we had to train our dense base on top to suit the binary classification. Thus, our remodeled CNN consists of 13 convolution layers and 2 dense layers. The number of filters in convolution layers increases (multiples of 64) as we go deeper into the network (from 64 to 512 filters). Each convolutional layer in the network is a 3 x 3 grid but has varying filters. It is expected that the filters will increase as we go deeper into the network because the level of abstraction (representation) and the number of features to extract increases. Also, generality reduces as the depth of the convolutional layers increases. This network applied the rectified linear unit (ReLU) activation with Maxpooling on the convolved outputs. Three main activations normally applied to the hidden layers are

sigmoid, hyperbolic tangent (tanh), and ReLU. Sigmoid and hyperbolic tangent activations have the vanishing gradient problem during backpropagation of errors which is corrected in ReLU and overall, ReLU gives the best model accuracy. After the last convolution, the output is flattened into a single stretch of neurons called the dense layer. For the fully connected layers, we have the first dense layer with 256 neurons and the second dense layer with 2 neurons. In the output layer, three activations are considered; we have the linear, the sigmoid, and the SoftMax. Sigmoid or SoftMax can be used for classification problems. However, in this study, SoftMax activation (a mathematical function that converts a vector of numbers into a vector of probabilities) was applied as it gave the best accuracy.

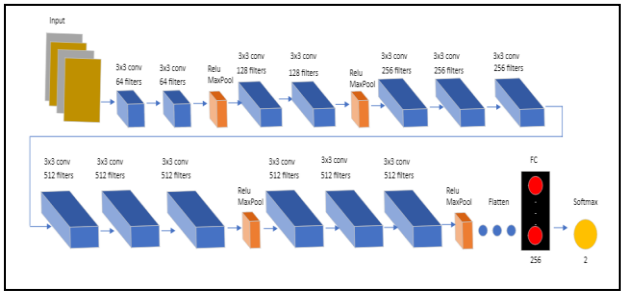

Figure 3. VGG16 Architecture.

VGG16 model comes prepackaged with Keras. The input image size was 224x224 pixels. We used the fine-tuning method and the steps involved adding the custom classifier base on top of the convolutional base; freezing the base network; training the part added, unfreezing some layers in the base network, and jointly training both these layers and the classifier. The weights are computed for only the convolutional layers. Also, after every convolution operation, the output image takes the depth of the filter convolved with. For instance, in the first convolutional layer, after the input image of dimension 224 x 224 x 3 is convolved with the filter of dimension 3 x 3 x 64, the resulting image dimension becomes 224 x224 x 64, where 64 is the number of filters used. Max-pooling layers have no parameters/weights. Max-pooling is a way to extract the maximum value representing a section of the output shape, usually defined by the number of strides. As stated, this layer does not compute any weights but reduces the output shape's dimension.

There are parameter specifications required to train the classifier base. This is done to increase the accuracy of prediction. The parameters include the cost function, the type of optimizer, and the learning rate. The cost function is the penalty when the predicted label deviates from the expected label. The focus is on the number of times when the model makes wrong class predictions, and so there is an inverse relation between the number of predictions and the cost, that is, the higher the number of correct predictions, then the lower the cost of the penalty, and vice versa. The cost function helps to update the weights of the network. There are many loss functions which include the mean squared error (MSE), mean absolute error (MAE), and mean squared logarithmic error (MSLE). However, there are cost functions for binary classifications which include the binary cross-entropy, and the hinge loss, and the former is the default cost function.

Machine learning optimizers help to adjust the weights of a neural network to minimize cost. Therefore, it significantly affects getting a good or a badly trained network. As time passed, gradient descent algorithms were used often. However, a major challenge was rightly setting the learning rate for training which is overcome in adaptive optimizers although some programs allow manual adjustment of the learning rate. Some examples of adaptive optimizers include Adaptive Gradient (Adagrad), Adaptive Delta (Adadelta), Root Mean Squared propagation (RMSprop), and Adaptive Moment Estimation (Adam) optimizers. We chose Adam because it combines the advantages of both the Adagrad and the Adadelta. Table II shows 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 Figures 1 and 2 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.

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 (10) in the following equation:

$$Accuracy = \frac{\sum TP+\mathrm{TN}}{\sum TP+FP+TN+FN} \qquad (10)$$

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 (11) as:

$$Recall = \frac{\sum TP}{\sum TP+FN} \quad (11)$$

Precision measures how the model found correct positive responses (TP) to the total number of positive responses. It is mathematically expressed by (12) as

$$Precision = \frac{\sum TP}{\sum TP+FP} \quad (12)$$

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:

$$F - score = 2 * \frac{precision*recall}{precision+recall} \quad (13)$$

According to [14], 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.

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 4:

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

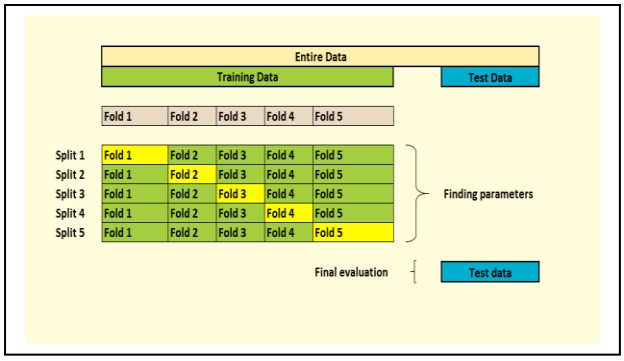

Figure 4. K-fold cross validation.

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 5 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.

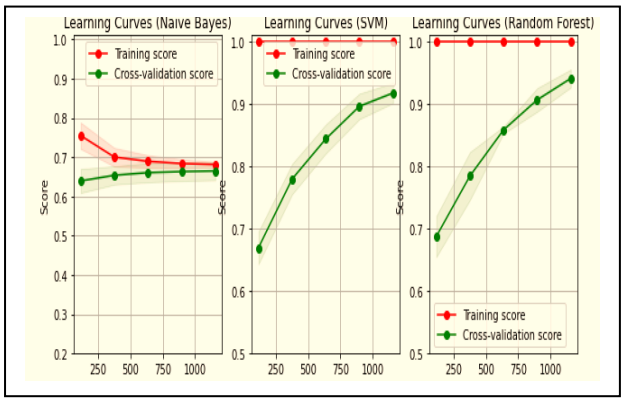

Figure 5. Learning curves for the tested classifiers.

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 6 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.

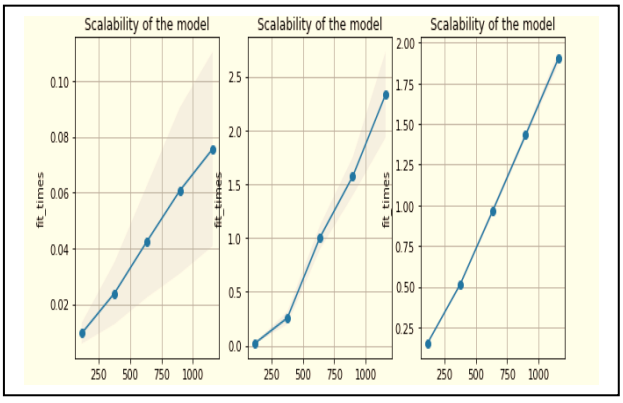

Figure 6. Scalability of the tested classifiers.

Figure 7 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 8, 9, and 10 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, whiles 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 each classifier. The positive sample represents images with drowsy labels, while the negative represents images with non-drowsy labels.

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.

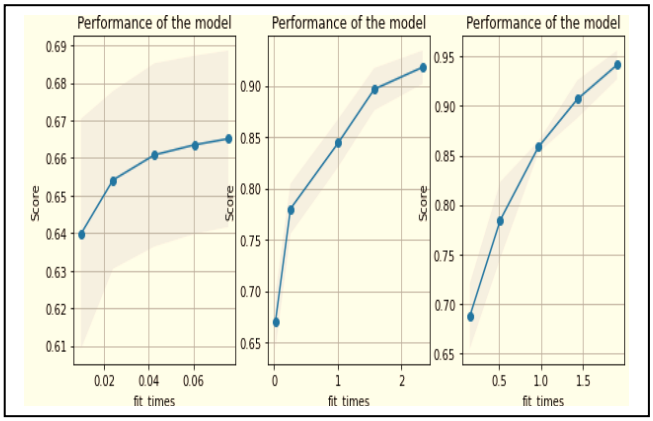

Figure 7. Performance of all three classifiers.

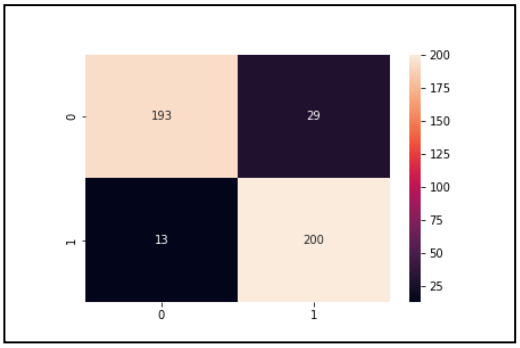

Figure 8. Confusion matrix for SVM.

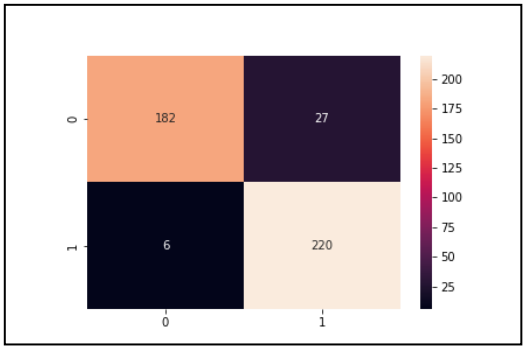

Figure 9. Confusion matrix for Random Forest.

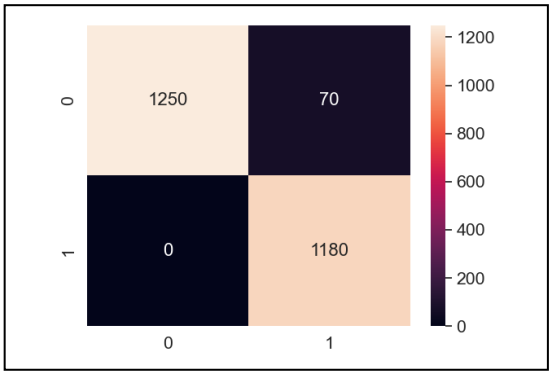

Figure 10. Confusion matrix for Naive Bayes.

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 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. PARAMETERS FOR TRAINING CLASSIFIER BASE.

Classifier	Accuracy	Precision	Recall	F-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 11. 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.

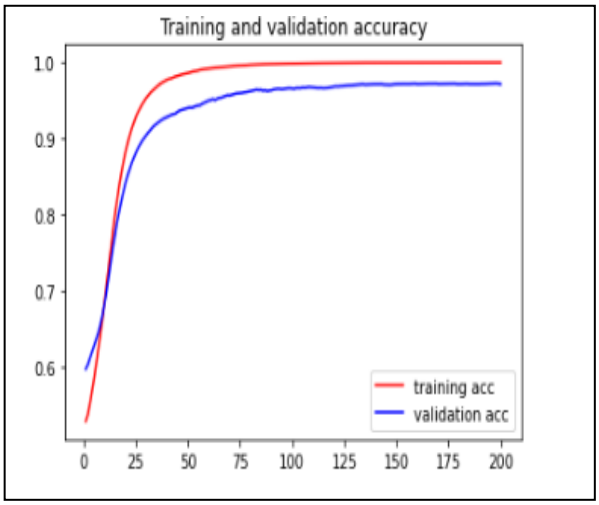

Figure 11. Plot for training and validation accuracy (200 epochs).

Figure 12 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.

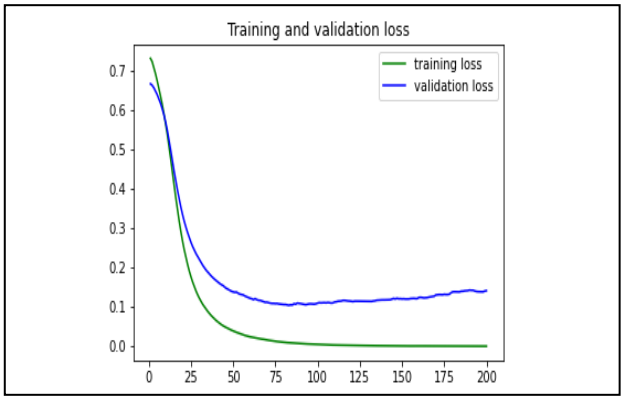

Figure 12. Plot for training and validation loss (200 epochs).

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 13. The left binary representation is the predicted label whereas that 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 Table III.

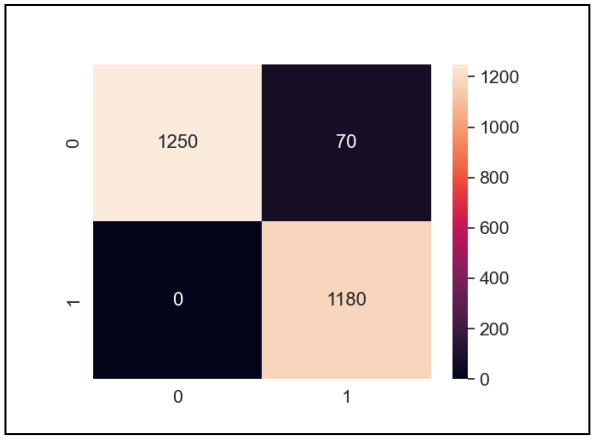

Figure 13. Confusion matrix for the neural network.

TABLE III. CONFUSION MATRIX FOR THE VGG NETWORK.

Total = 2500		**Actual State**	
		Drowsy 1250	**Not drowsy** 1250
Predicted State	**Drowsy** 1320	1250	70
	Not drowsy 1180	0	1180

The accuracy, precision, recall, and F-score are computed for this network and summarized in Table IV. 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 IV. PERFORMANCE OF VGG16 NETWORK.

Classifier	**Accuracy**	Recall	Precision	F-score
VGG16	97.20	100	94.70	97.28

4. CONCLUSION AND FUTURE WORK

This research uses three traditional machine learning classifiers and a deep neural network for drowsiness detection. In previous literature, the EAR and MOR values were recorded from images and used as input for the classifiers. However, in this study, the usage of image pixels as classifier input values was considered, which provided good accuracy. The former heavily depends on the recorded values being accurate. The image pixel method is novel and gives a good result as the geometry of the face determines the input pixel values. Again, the neural network is effective in accurately classifying drowsiness with a very high accuracy. The deep neural network outperformed all traditional methods with an accuracy of 97. All methods are non-invasive and cost-effective. These algorithms can be integrated into automobile dashboards for easy detection. In the future, the number of image data can be increased for testing the models. In the future, other state-of-the-art networks can be considered, and comparisons made while considering the trade-off between the accuracy, computational complexity, time of execution, and implementation cost.

ACKNOWLEDGMENT

The authors acknowledge the support of the Louisiana Transportation Research Center in supporting the work performed in this paper through LTRC Project Number 22-2TIRE. The authors would also like to thank the support of Southern University and A&M College and Morgan State University 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), Bahrain, Oct. 2022, pp. 591-596.

[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," in IEEE Access, vol. 7, pp. 61904-61919, 2019, doi: 10.1109/ACCESS.2019.2914373.

[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), Shimla, India, Nov. 2019, pp. 108–113, doi: 10.1109/ICIIP47207.2019.8985844.

[4] W. Kong, L. Zhou, Y. Wang, J. Zhang, J. Liu, and S. Gao, "A system of driving fatigue detection based on machine vision and its application on smart device," Journal of Sensors, vol. 2015, 2015, doi: 10.1155/2015/548602.

[5] 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), Doha, Qatar, Feb. 2020, pp. 237–242, doi: 10.1109/ICIoT48696.2020.9089484.

[6] Y. Chellappa, N. N. Joshi, and V. Bharadwaj, ``Driver fatigue detection system," in Proc. IEEE Int. Conf. Signal Image Process. (ICSIP), Aug. 2016, pp. 655_660.

[7] 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, Aug. 2017.

[8] 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), Doha, Qatar, Feb. 2020, pp. 237–242, doi: 10.1109/ICIoT48696.2020.9089484.

[9] 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), Chennai, India, Feb. 2015, pp. 262–266, doi: 10.1109/ICCCT2.2015.7292757.

[10] W. Deng and R. Wu, "Real-Time Driver-Drowsiness Detection System Using Facial Features," IEEE Access, vol. 7, pp. 118727–118738, 2019, doi: 10.1109/ACCESS.2019.2936663.

[11] N. Kumar and N. C. Barwar, "Analysis of Real-Time Driver Fatigue Detection Based on Eye and Yawning," Int. J. Comput. Sci. Inf. Technol., vol. 5, no. 6, pp. 7821–7826, 2014.

[12] U. S. Reddy, V. Rami, R. Chirra, S. Reddy Uyyala, V. Krishna, and K. Kolli, "Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State," 2020, doi: 10.18280/ria.330609.

[13] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, "YawDD: A Yawning Detection Dataset", Proc. ACM Multimedia Systems, Singapore, March 19 -21 2014, pp. 24-28. DOI: 10.1145/2557642.2563678.

[14] 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), Noida, Aug. 2018, pp. 1–6, doi: 10.1109/IC3.2018.8530651.

[15] K. Simonyan and A. Zisserman. (2014). ''Very deep convolutional networks for large-scale image recognition.'' [Online]. Available: https://arxiv.org/abs/1409.1556

[16] Willson Meli, Fred Lacy, and Yasser Ismail "Video-Based Automated Pedestrians Counting Algorithms for Smart Cities" International Journal of Computing and Digital Systems (IJCDS), 2020.

[17] Y.Ismail, M. Hammad, M. Darwichand, and W. Elmedany "Homeland Security Video Surveillance System Utilizing the Internet of Things (IoT) for Smart Cities" IET Computers & Digital Technique journal, Volume 15, Issue 4, Pages: 241-319, 04 April 2021.