OnlineDC: Leveraging Temporal Driving Behavior to Facilitate Driver Classification

Hashim Abu-gellban*, Yu Zhuang*, Long Nguyen†, Fang Jin‡, Zhenkai Zhang∓
*Department of Computer Science, Texas Tech University
†Computer Science and Data Science, Meharry Medical College
‡Department of Statistics, George Washington University
∓Computer Science Division, Clemson University
hashim.gellban@ttu.edu, yu.zhuang@ttu.edu, hlnguyen@mmc.edu, fangjin@gwu.edu, zhenkai@clemson.edu

***Abstract*—Driver classification is used recently for vehicle anti-burglary and fake driver accounts based on driving behavior. Anti-burglary is a challenging problem as it leans on external devices to defend against vehicle theft. Several researchers analyzed the driving behavior to identify drivers, but they faced several challenges to produce a stable model for the cold start problem and for medium-long sequences. In addition, some approaches had an unpleasant performance when the action space increased (> 2 drivers). In this paper, we propose a novel approach named OnlineDC (Online Driver Classification), which leverages temporal driving behavior to identify a human subject behind the wheel. Our method utilizes the Gated Recurrent Unit (GRU) and the ResNet with the Squeeze-Excite blocks (SE) to analyze the long-short term patterns of driving behaviors. Moreover, we fostered the performance by building and applying the Feature Generation (FG) algorithm to extract spectral, temporal, and statistical features from the sensing data of vehicles. We conducted extensive experiments to show how our approach outperformed state-of-the-art baseline methods. The results also showed that our solution could resolve the cold-start problem for short patterns.**

***Index Terms*—Driver Classification, Cybersecurity, Deep Learning, Neural Network, Internet of Things.**

I. Introduction

Vehicle burglary is ubiquitous around the world, and numerous victims of theft have suffered the loss of their properties. For instance, the Federal Bureau of Investigation (FBI) announced in fall 2019 that there were approximately 748,841 thefts of motor vehicles in the United States last year[1]. Several approaches were used to address the burglary issue.

Most early studies in the driver identification domain mainly focused on biometric approaches. These technologies have key hindrances such as low accuracy that negatively impact implementing large scale applications, economic reasons for the car manufacturers to implement them, and privacy issues [1], [2].

Current cars' security systems have some problems. For example, vehicle manufacturers embed a security device containing GPS in a secret part of a vehicle to send its geolocation [3], to facilitate finding it when lost. However, the device does not prevent a burglary in advance, and numerous drivers might consider GPS as an invasion of their privacy. Furthermore, car manufacturers used onboard systems inside vehicles for the anti-burglary purpose. However, the controller exposed an opportunity to reverse engineer the technology to steal the vehicle. For instance, a BBC report [2] mentioned that even though some cars were implemented onboard systems for securing cars where a driver had a fob to start a car, thieves targeted some of these cars and remotely stole them. As a result, many car insurances forced drivers to have external devices.

Sensors embedded in vehicles are essential for modern control systems for driving safety. These data can be extracted through the OBD (On-Board Diagnostics) port. The original use of these data was for maintenance. Vehicular Ad-hoc Networks (VANETs) utilize these sensor data and the vehicle's communication from the surrounding other vehicles' sensors and communication to produce several applications. Recently, researchers started using data from car sensors for the driver classification problem and employing the highest similarity for driver verification [4].

Using driving behaviors (i.e., sensing data) is another approach to detect car burglary by applying Deep Learning approaches and supervised tree-based/ lazy classification algorithms to identify drivers. Researchers made different use of such data. For example, Virojboonkiate *et al.* [5] extracted 100 sensing data and generated 100 histograms to be trained in a neural network (NN) with accuracy (below 76%). While, Ferreira *et al.* [6] used datasets with a short time window ($\leq$ 8 seconds) applying the Multi-Layer Perceptron (MLP). Likewise, Jeong *et al.* [7] analyzed driver behavior for a medium to a long time window (from 75 to 4860 seconds) and only 4 drivers, using the three Conv1D layers. Moreover, Kwak *et al.* and Martinelli *et al.* [8], [9] employed Random Forest, XGBoosting, and J48, but the performance in the proposed approach by Kwak *et al.* [8] dramatically declined after the action space increased (i.e., increasing the number of drivers from 2 to 3, 4, and 5). Another approach was suggested by Martinelli *et al.* [10], who applied the KNN, SVM, Gaussian Process, Decision Tree, Random Forest, AdaBoost, Naive Bayes, and Logistic algorithms. They suggested using the Lazy algorithm (KNN) to classify drivers according to their driving

[1]https://ucr.fbi.gov/crime-in-the-u.s/2018/crime-in-the-u.s.-2018/topic-pages/motor-vehicle-theft (2019), [Last Access 8/3/2021]

[2]https://www.bbc.com/news/technology-29786320 (2014), [Last Access 8/3/2021]

behaviors. The main issue of KNN is its lazy nature, which makes its algorithm not scalable for online driver applications with a large number of driver behaviors as KNN needs a longer time to evaluate the test examples. Furthermore, Rahim *et al.* [11] used GPS data to train the Random Forest algorithm. Whereas, GPS may be considered by several people as an invasion of their privacy. However, all these previous works had several limitations or issues. The neural networks were shallow or not suitable to learn from driving behaviors. Also, some approaches may not extract adequate features to enhance the identification model, and some of them did not extract any feature from sensing data. These approaches suffered from providing high accuracy in the cold start problem when given a little driving behavior, and for long term patterns. It is essential for the classification model to generate a stable model with different lengths of senor sequences. Moreover, Kwak *et al.* [4] suggested an alarm system that would inform the owner of the vehicle and a vehicle monitoring company in case of a burglary event based on the driving behavior. However, the response time for the alarm notification is crucial (e.g., 10 seconds).

Ride-sharing services like Uber and Lyft on the taxi industry may need long lengths of signals to identifying fake driver accounts for security and administration purposes. For example, Uber created 60 thousand jobs in China in May 2015[3], where several drivers attempted to manipulate the system to get more bonuses through having several accounts as mentioned in an article in Quartz[4]. A manual monitoring approach made it hard to find fake accounts among all Uber drivers in China or any other country. An automatic stable driver classification models using machine learning techniques can help to address this problem.

A stable driver classification model has to address these two requirements (i.e., providing high performance identification for a short and a long time series) for a reasonable number of drivers (e.g., 10). Our proposed approach has high accuracy between 90% and 97%, for different driving intervals from 10 seconds to 25 minutes having 14 drivers.

To address these problems of short and long window size as well as the size of action space keeping high performance, we propose OnlineDC (i.e., Online Driver Classification), which is a data-driven approach. OnlineDC is based on a Deep Learning model to identify drivers from driving behaviors while keeping high accuracy. We first decrease the data dimensionality using two signal selection algorithms (Chi-square [12], [13], and Univariate linear regression [14]). After that, we generate 82 features using our proposed feature generation algorithm (FG) which has fixed the input space for any length of multivariate time series (signals) to increase the scalability of OnlineDC. As a result, we trained our classifier only between 43 and 46 seconds for different signals' lengths. Finally, we use our new neural network (MGRU-ResSE) to identify drivers with high performance. MGRU-ResSE utilizes Gated Recurrent Unit (GRU) and ResNet with the squeeze and excite blocks.

Our main contributions in this paper are:

- We propose a model that can classify drivers with high accuracy. The model combines Gated Recurrent Unit (GRU) and ResNet with the squeeze and excite blocks in the multivariate time series domain. GRU reduces the training time and learns temporal patterns as in LSTM. The ResNet with the squeeze_excite architecture is applied to extract more latent features to improve the model while addressing the vanishing gradient problem.
- We employ our Feature Generation (FG) algorithm to extract spectral, temporal, and statistical features from sensing data. FG improves the performance for short and long driving patterns as well as helps to stabilize the model. FG unifies the size of the input space for different window lengths to increase the scalability of our OnlineDC.
- Our numerous experiments manifest the capabilities of our proposed approach. The results show that OnlineDC is promising and outperforms baseline methods for different sequence lengths, including addressing the cold start problem.

II. Problem Formulation

As the modern vehicle has numerous sensors, we intend to learn the driving behavior leveraging data from these sensors. We formulate the driver classification problem, as follows:

The input: The input is K sensors for D drivers, denoted by $(B, Y) = (B^1, y_1), ..., (B^N, y_N)$, where B^i is the temporal driving behaviors of the driver y_i where $y_i \in [1, D]$ is the class label and N is the number of driving behaviors. B^i is MTS (Multivariate Time Series) and $B^i = [S^1, ..., S^K]$, where S^k is a sequence with length T representing the k^{th} signal.

Problem definition: Given B^i, we select M signals from B^i sequences, where $M < K$. Next, we extract F features from B^i to generate $\{X^1, ..., X^F\}$, where $j \in [1, F]$ and $|X^j|$ is equal to the number of the selected signals (M). Our target function for the driver classification problem is finding the function f that best identifies the driver y_i:

$$y_i = f(X^1, X^2, ..., X^F) \tag{1}$$

III. Related Work

A. Traditional Classification Algorithms

Kwak *et al.* [8] and Ezzini *et al.* [15] analyzed the driver behaviors through a multi-step process using data collected from vehicle sensors and applying the Extra-Trees, Decision Tree, Random Forest, XGBoost, and SVM algorithms. Furthermore, Enev *et al.* [2] used the CAN network dataset that contained 13 signals (e.g., fuel consumption, throttle position, acceleration, brake, and steering wheel) for a short period (i.e., between 200 milliseconds and 15 seconds). They employed a lazy algorithm (KNN) and other traditional classification algorithms (i.e., Random Forest, SVM, Naive Bayes) to discover the diver class label based on driving behaviors. Ferreira

[3]https://www.usatoday.com/story/tech/columnist/stevenpetrow/2016/10/12/fake-uber-drivers-dont-become-next-victim/91903508/

[4]https://qz.com/423288/fake-drivers-and-passengers-are-boosting-ubers-growth-in-china/

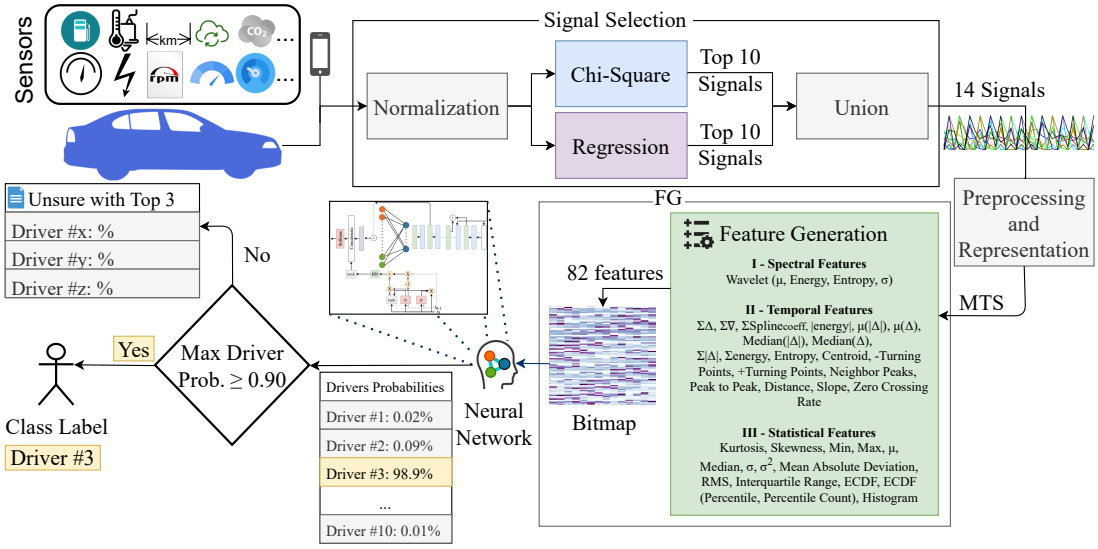

Fig. 1. The overview of our data-driven technique (OnlineDC).

et al. [6] used datasets with short driving behavior (time window was between 4 and 8 seconds) and applied SVM, Random Forest, and Bayesian Network algorithms. Martinelli *et al.* [10] selected two signals (i.e., the torque of friction and the intake air pressure) for the analysis. They applied the Nearest Neighbor, SVM, Gaussian Process, Decision Tree, Random Forest, AdaBoost, Naive Bayes, and Logistic algorithms to discriminate legitimate and illegitimate drivers. They suggested using the KNN (Lazy algorithm) to find the driver according to their driving behavior. However, the evaluation time of KNN for large data takes a long time as there is no training phase. Rahim *et al.* [11] proposed a driver identification model using the Random Forest algorithm and based on GPS data. Using GPS impacts driver privacy which several drivers may refuse to use the proposed approach. Rettore *et al.* [16] used PCA (Principal Component Analysis) to reduce the input space [17] and applied an auto machine learning (AutoML) tool called TPOT [18], [19]. The main issue of PCA is that it does not consider the target, which makes PCA not fit for many traditional classification algorithms [20]. Whereas, there are other attribute selection algorithms suitable for the classification problem like Chi-square and Univariate linear regression. Rettore *et al.* found that the best-chosen algorithm by TPOT was the Extra-Tree. Tahmasbi *et al.* [21] used smartphone sensors and applied a gradient boosting tree (GBT) classifier.

B. Deep Learning Approaches for Multivariate Time Series Classification

Several researchers used Deep Learning for the driving behavioral classification. Ferreira *et al.* [6] also used Artificial Neural Network (i.e., a simple neural network layer called the Multi-Layer Perceptron (MLP)), which may not produce a model with high performance. Additionally, Jeong *et al.* [7] analyzed driver behavior for a medium to a long time window (from 75 to 4860 seconds) and only 4 drivers. They used 2 to 10 sensors of the followings: Acceleration Pedal, Brake Pedal, Velocity, Throttle Valve Angle Value, Engine RPM, Longitudinal Acceleration, Lateral Acceleration, Yaw Rate, Steering Wheel Angle, and Steering Wheel Rotation Speed. They designed a neural network model containing three Conv1D layers, where each layer was followed by Max Pooling. In addition, we implemented SFG to reduce B^i and trained on MGRU-FCN (Multivariate GRU-FCN) in our previous paper [22]. However, the model did not tackle the cold start problem, while Chen *et al.* [1] developed an autoencoder (DNCAE) without decreasing the size of B^i. [23], [24] applied ResNet-50 (i.e., 1 dense layer and 49 Convolutional Neural Network layers "CNNs" where every 3 CNN layers have a shortcut or residual operation to the consecutive CNN layers) and GRU with 69% and 90% accuracy of Top-1 and Top-5 [23], and with 71% and 92% accuracy of Top-1 and Top-5 [24]. Accuracy top-X represents the success percentage if X drivers or more among other drivers (5-25) have the highest probability at the journey level. However, the accuracy of Top-1 and Top-5 for 7 drivers was less than 45%. ResNet-50 has only CNN layers without any batch normalization, activation function (e.g., ReLU), or advanced neural network block like Squeeze-Excite [25] to increase the performance of the classification model. [23], [24] did not use feature generation techniques (e.g., statistical or temporal features) to reduce the input size keeping high performance. The lack of reducing the input state needs more time to train the neural network containing a large number of parameters (ResNet-50).

IV. Online Driver Identification Framework

Fig. 1 represents our general approach from reprocessing phases until identifying the driver class label using Deep Learning. There are four steps: *signal selection, data prepro-*

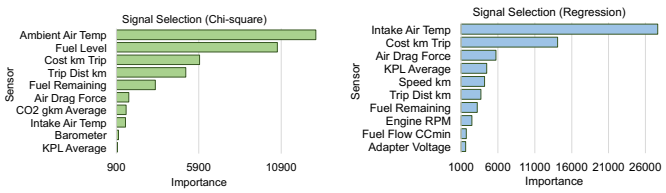

Fig. 2. The selected signals using Chi-square and Regression.

cessing and representation, feature generation, and *building the driver identification model*.

A. Signal Selection

Sensing data from a vehicle should be encrypted using several network security approaches, including RSA and CDMI [26]. We do not use the geolocation data (i.e., GPS) to protect personal privacy in our approach. We first scale our dataset using the Min-Max normalization for each signal (s), to convert the signal points between 0 and 1 that helps the Chi-square algorithm, which takes only non-negative variables. Let $x_{\min_s}$ and $x_{\max_s}$ are the minimum and maximum values in the signal s for all examples. The normalization formula for only the signal selection process is as follows:

$$x_s^{'} = \frac{x_s - x_{\min_s}}{x_{\max_s} - x_{\min_s}} \tag{2}$$

We apply two different methods to select the top 10 sequences based on signals' importance (Fig. 2). After that, we unify the top signals to extract only 14 distinct signals and pass them for further preprocessing tasks.

There are other methods to reduce the input space, such as PCA (Principal Component Analysis) [17], [27] and ANOVA [28]. PCA is computed using only the input data without the class labels that make PCA more suitable for unsupervised learning algorithms rather than supervised algorithms [20]. In our experiments, ANOVA did not help our classification approach to produce a stable model with high performance for different time series lengths. Thus, we employ only Chi-square and Univariate linear regression algorithms to select the signals and reduce the input space in our driver identification framework.

1) Chi-square (χ^2): This algorithm is commonly used to select attributes for the multivariate time series domain [12], [29]. χ^2 test computes the dependency between a non-negative attribute and a class label [13]. We select the top 10 highest χ^2 values to reduce the dimensions of the driving behaviors B. The formula of χ^2 is:

$$\begin{aligned} \chi_k^2 &= \frac{(O_k - E_k)^2}{E_k} \\ O_k &= \sum_{i=1}^{N} x_{i,k}\ y_i \\ E_k &= \overline{y} \sum_{i=1}^{N} x_{i,k} \end{aligned} \tag{3}$$

Algorithm 1: The feature generation algorithm "**FG**".

function feature_generation (MTS)
Input : MTS $= X \in \mathbb{R}^{N \times M \times T}$
Output: FG_MTS $= X^{'} \in \mathbb{R}^{N \times F \times M.}$
1. transpose MTS per example (y): MTST $=$ MTS^T.
2. generate F features from MTST for each example: FG2D = generate_features(MTST, F).
3. replace ∞ values of features by its μ: FG2D = replace_infinity(FG2D).
4. scale the data per each feature f to generate a bitmap: $x_f^{'} = \frac{x_f - x_{\min_f}}{x_{\max_f} - x_{\min_f}}$.
5. reshape FG2D into MTS: FG_MTS=reshape(FG2D, N, F, M)
return FG_MTS

2) Univariate linear regression: It is a scoring function to select important variables. First, we compute the correlation between the regressor and the class labels for each signal ($k \in \{1..K\}$) [30] as follows:

$$corr_k = \frac{\sum_{i=1}^{N} (x_i - \overline{x}) * (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \overline{x})^2 * \sum_{i=1}^{N} (y_i - \overline{y})^2}} \tag{4}$$

where $\overline{x}$ and $\overline{y}$ are the mean value of the signal x. Also, y represents a class label. We convert the correlation to:

$$F_k = \frac{\text{Corr}_k^2 * (n-2)}{1 - \text{Corr}_k^2} \tag{5}$$

where n is the number of examples.

Using the two algorithms, there are several signals which are selected by both algorithms as shown in Fig. 2. The common signals are Cost km Trip, Trip Dist km, Fuel Remaining, Air Drag Force, Intake Air Temp, and KPL Average. Whereas, the Ambient Air Temp, Fuel Level, CO2 gkm Average, and Barometer signals are only selected by the Chi-square algorithm. The Ambient Air Temp is used to distinguish the Intake Temp values impacted by external air conditions or other factors (e.g., the engine's temperature) since our model computes interdependencies between the channels. In the dataset, this temperature has the same value in each car. Additionally, Speed km, Engine RPM, Fuel Flow CCmin, and Adapter Voltage are only chosen by the regression approach.

B. Data Preprocessing and Representation

We represent the 2-D dataset into MTS with a fixed length (e.g., 20 and 1500). We shuffle and randomly split the dataset into training and testing datasets. We also make sure that each sequence in MTS is in chronological order. Then, we fill the gap in data by applying forward and backward interpolation (PCHIP [31]).

C. Feature Generator (FG)

Algorithm 1 describes how we extract the features from MTS using our FG. Step 1, we reshape MTS into $X \in$

TABLE I
THE SET OF THE GENERATED FEATURES

Domain	Feature	Formula/ Source
Spectral	Wavelet Absolute Mean	$\lvert\mu_{\text{wavelet}(S)}\rvert$
	Wavelet Energy	[32]
	Wavelet Entropy	[33]
	Wavelet Standard Deviation	$\lvert\sigma_{\text{wavelet}(S)}\rvert$
Temporal	Sum Cubic Spline Coefficients	$\sum \text{CSC}(S)$
	Sum Gradient (Squared)	$\sum_{t=1}^{T} \nabla S_t^2$
	Sum Differences (Squared)	$\sum_{t=2}^{T} \Delta S_t^2$
	Absolute energy	$\sum_{t=1}^{T} S_t^2$
	Mean Absolute Differences	$\mu(\lvert\Delta(S)\rvert)$
	Mean Differences	$\mu(\Delta(S))$
	Median Absolute Differences	$\text{median}(\lvert\Delta(S)\rvert)$
	Median Differences	$\text{median}(\Delta(S))$
	Sum of Absolute Differences	$\sum_{t=2}^{T} \lvert\Delta(S_t)\rvert$
	Total Energy	$\frac{\sum_{i=1}^{T} S_i^2}{T}$
	Normalized Entropy	$\frac{-\sum_{u \in S} P(u)\ log_2 P(u)}{\log_2 \lvert S\rvert}$
	Centroid	$\frac{\sum_{i=1}^{T} t_i \times S_i^2}{\sum_{i=1}^{T} S_i^2}$
	Negative Turning Points	$\text{count}(\Delta S_t < 0$ and $\Delta S_{t+1} > 0)$
	Positive Turning Points	$\text{count}(\Delta S_t > 0$ and $\Delta S_{t+1} < 0)$
	Neighbourhood Peaks	[34]
	Peak to Peak	$\lvert\max(S) - \min(S)\rvert$
	Signal Distance	$\sum_{t=2}^{T} \sqrt{1 + \Delta S_t^2}$
	Slope	m, where $y = m\ x + b$
	Zero Crossing Rate	$\sum_{t=2}^{T}(\lvert\Delta f(S_t)\rvert)$, where $f(S_t) = 1$ if $S_t > 0$ otherwise 0
Statistical	Kurtosis	$\text{kurtosis}(S)$
	Skewness	$\text{skewness}(S)$
	Minimum	$\min(S)$
	Maximum	$\max(S)$
	Mean	μ
	Median	$\text{median}(S)$
	Standard deviation	σ
	Variance	σ^2
	Mean Absolute Deviation	$\frac{\sum_{i=1}^{T} S_i^2 - \mu(S)}{T}$
	Root Mean Square	$\sqrt{\frac{\sum_{i=1}^{T} S_i^2}{T}}$
	Interquartile range	$Q_{3^{rd}} - Q_{1^{st}}$
	ECDF	[35]
	ECDF Percentile	$\text{Max}_p\text{sort}(S)$
	ECDF Percentile Count	$\text{count}(\text{sort}(S) \leq \text{threshold})$
	Histogram	$\sum_{k=1}^{\#\text{bins}} \text{Histogram}_k$

$\mathbb{R}^{N \times M \times T}$. Let $S \in B^i$ denote a univariate time series (i.e., a signal) in the driving behavior B^i, and wavelet is the Continuous Wavelet Transform (CWT). For every signal (step 2), we generate 82 features briefed in Table I.

1) Spectral Features: We generate 4 different types of spectral features, where each spectral feature has a width (a) between 1 and 9. Therefore, we extracted 28 spectral features in total, where 9 values for each feature except the Wavelet Entropy feature.

2) Temporal Features: We compute the gradient for each subsequence using the finite second-order central differences of the interior values of a signal (S) [36], [37]. For the first and last elements, we employ the second-order forward and backward differences, since there is no point before the first point and after the last point. For ECDF percentile and count, p is equal to 20% and 80%, while the threshold is set to 0.2 and 0.8.

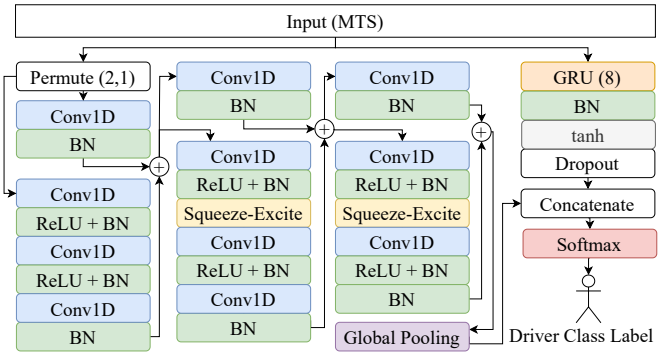

Fig. 3. The architecture of our proposed neural network (MGRU-ResSE) used in OnlineDC, to identify the driver.

3) Statistical Features: We compute 35 distinct statistical types. Whereas, the last 4 features are compound. ECDF produces 10 values (i.e., features), ECDF Percentile generates 2, ECDF Percentile Count has 2 values, and the Histogram contains 10 values (i.e., features). Furthermore, the term $Q_{1^{st}}$ is the first quartile (i.e., percentile for $q = 25$), and $Q_{3^{rd}}$ is the third one (i.e., $q = 75$). The q^{th} percentile of signal S is computed by sorting S to get S' and then computing the q^{th} ranked value in S'. In case that the normalized ranking does not match, q the values, distances of the two adjacent points, and the interpolation parameter are used to calculate the percentile.

We replace infinity values for every generated feature into the mean of the feature. To generate a bitmap, we use Min-Max normalization per feature (f):

$$x_f^{'} = \frac{x_f - x_{\min_f}}{x_{\max_f} - x_{\min_f}} \tag{6}$$

where $x_{\min_f}$ and $x_{\max_f}$ are the minimum and maximum values in the feature f in all examples. The normalization produces values between 0 and 1 to enhance the performance of neural networks by removing bias to large values [38]. Finally, we reshape the 2-dimensional feature into MTS (3D).

D. Driver Identification

We design a new neural network architecture (MGRU and ResNet with Squeeze-Excite called "MGRU-ResSE") to identify the driver class label shown in Fig. 3.

1) Gated Recurrent Units Network (GRU): GRU addresses the LSTM's vanishing gradient problem for a long time series [39]. The update gate solves the vanishing problem by learning historical information to flow future data. GRU learns the long-short pattern as LSTM, while GRU is faster than LSTM in learning [40], [41]. It is employed to extract high-level features to improve performance [42]. GRU contains

weight matrices (W and U), where the equations for GRU are:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1}) \\ \tilde{h}_t &= \tanh(W x_t + U(r_t \odot h_{t-1})) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned} \tag{7}$$

where $z_t, r_t, \tilde{h}_t$ and h_t are the update gate, the rest gate, the current memory gate, and the final memory gate at time t, respectively. The elementwise (Hadamard) multiplication is denoted by $\odot$. We utilize the tanh activation function followed by a dropout which helps in preventing overfitting [43].

2) ResSE: ResSE is Residual Neural Network with Squeeze-and-Excite block. Residual Neural Network (ResNet) is a collection of convolution blocks (Conv1D), the activation functions (ReLU), and the batch normalization layers (BN). The convolution blocks are efficiently used in discovering the semantic segmentation of pictures [44]. The formula of the basic convolution block is:

$$y = W \otimes x + b \tag{8}$$

$\otimes$ denotes the convolution operator. We employ eleven 1-D convolution layers (Conv1D) in our neural network. ReLU (Rectified Linear Units) is a low computation cost. The Batch Normalization layer (BN) helps to reduce the learning time and to support the generalization. The formulas for ReLU and BN are:

$$\text{re} = \text{ReLU}(y) \tag{9}$$

$$\text{bn} = \text{BN}(x) \tag{10}$$

where x is the output of Conv1D (y) or the ReLU layer (re).

The residual operation (h^{res}) is a shortcut connection to add the outputs of two batch normalization layers (h^{BN_1} and h^{BN_2}), where "+" is the residual operator. This shortcut improves the performance of our driver classification model by addressing the vanishing gradient problem [45].

$$h^{\text{res}} = h^{\text{BN}_1} + h^{\text{BN}_2} \tag{11}$$

We employ squeeze-and-excite block (SE), to discover interdependencies between the channels [46]. SE is built for any transformation $F_{tr} : X \rightarrow U, X \in \mathbb{R}^{H' \times W' \times C'}$, and $U \in \mathbb{R}^{H \times W \times C}$. $V = [v_1, ..., v_C]$ is a set of kernels. F_{tr} is represented by $U = [u_1, ..., u_C]$ is the output of F_{tr} where $u_c = v_c \otimes X = \sum_{s=1}^{C'} V_c^s \otimes X^s$ and V_c^s is a 2D kernel. $z \in \mathbb{R}^C$ is used to decrease the dimensions of U by $H \times W$: $z_c = F_{sq}(u_c) = \frac{1}{(H \times W)} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$. The excite operation is employed to aggregate Z by ReLU followed by the sigmoid (σ) activation function: $s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 ReLU(W_1 z))$, where W_1 and W_2 are the dimension layers. The output ($\tilde{X} = [\tilde{x}_1, ..., \tilde{x}_C]$) of SE is scaled the U with the activation: $\tilde{x}_c = F_{scale}(u_c, s_c) = u_c . s_c$.

3) Combining GRU and ResSE: The concatenation of GRU and ResSE neural networks is shown in Fig. 3. This combination enhances the model performance. This combination is defined by:

$$\begin{aligned} h^c &= h^g \oplus h^d \\ y &= \text{softmax}(h^c) \end{aligned} \tag{12}$$

where $\oplus$ is the concatenation operator, and h^c is the output of concatenating the global pooling (h^g) and the dropout (h^d). The driver class label is predicted by the softmax activation. The concatenation provides the softmax activation function with the latent features generated from the ResNet with squeeze_and_excite blocks (NN1), and the long short patterns from GRU (NN2). Using both neural networks (NN1 and NN2) produces better performance than using a single one (NN1 or NN2).

4) The Loss Function: We employ the categorical cross-entropy loss function (called the softmax loss) to train our classification model since this loss function is commonly used in neural network after the softmax activation function [25], [47], [48]. The formula of the loss function is as follows:

$$\text{Loss} = -\sum_{d=1}^{\text{D}} y_d \cdot \log \hat{y}_d \tag{13}$$

where y_d represents the probability of the actual d^{th} class label (i.e., the actual binary vector of a training driving behavior), $\hat{y}_d$ denotes the probability of the predicted d^{th} class label (i.e., the predicted softmax probability distribution over D drivers for the training driving behavior) generated by the classification model, and D is the number of drivers in the model (i.e., the number of class labels). To train the neural network, any class label of an example is converted to a vector of zeros and a one (i.e., the one-hot vector). The loss function sums only the loss values for the actual class label (mutually exclusive) in the vector since other values are zeros.

We check the maximum probability of class labels against a threshold (e.g., 0.90) to verify the results since classifiers select one class label even when they are not sure. Therefore, if the maximum probability is larger than or equal to a threshold (≥ 0.90), we show the related class label; otherwise, we present the unsure report containing the top drivers.

V. Competing Approaches

Random Forest, XGBoost, SVM, and MLSTM-FCN are the baselines methods for comparison. For the first three baseline methods, we use FG2D (i.e., 2D) after step 4 in Algorithm 1 instead of FG_MTS (i.e., 3D), to be suitable for these methods. The state-of-the-art techniques are:

Random Forest: Random Forest generates numerous decision trees randomly during training the classification model. After that, it employs bagging approaches to create an optimized model. The random decision forests algorithm is commonly used for classifying drivers based on their behaviors [8], [15], [49].

XGBoost: In 2016, Chen developed the eXtreme Gradient Boosting (XGBoost) algorithm [50]. XGBoost is a scalable

TABLE II
OVERALL PERFORMANCE SUMMARY OF OUR APPROACH AGAINST THE BASELINE ALGORITHMS ($T = 10$).

Algorithm	Acc.	Prec.	Rec.	F1-score	Kappa
Random Forest	0.78	0.75	0.78	0.74	0.73
XGBoost	0.75	0.71	0.75	0.73	0.70
SVM	0.41	0.23	0.41	0.29	0.19
MLSTM-FCN	0.84	0.84	0.84	0.82	0.80
OnlineDC	**0.91**	**0.91**	**0.91**	**0.90**	**0.89**

tree boosting approach by applying the gradient boosting machine [51] and running in parallel which decreases the training time. It was employed in numerous cybersecurity domains [52]–[54].

SVM: The algorithm of SVM converts the input space into a higher-dimensional space based on the kernel function, to discover support vectors that maximize margins between classes [55], [56]. SVM has been used in several research papers for the driver classification problem [2], [8].

MLSTM-FCN: Karim *et al.* designed the Multivariate LSTM-FCN (MLSTM-FCN) for classifying multivariate time series [25], [57].

VI. EXPERIMENTS AND RESULTS

Instead of reporting the best value obtained, we ran each algorithm 10 times and took the average of each metric in our experiments, unless we mention otherwise. Additionally, we trained all classifiers using the generated features by FG except for some experiments in Section VI-D where we ran our approach without the Feature Generation algorithm for comparison. Furthermore, we trained the neural networks for 30 epochs in all experiments.

A. Datasets Description

The dataset [58] has eight men and six women driving Renault and Hyundai cars. The ages of Renault's drivers are between 25 and 61, while the ages of other vehicle's drivers are between 20 and 53. The experiments of Renault and Hyundai had 40 trips (28 hours per trip) and eight trips (3 hours per trip), respectively. The dataset contains 38 sensors from mobile phones and vehicles (OBD-II). Furthermore, most previous research works in the related work (Section III) used datasets with similar sizes or less such as Kwak *et al.* [8] and Jeong *et al.* [7]. In our previous research [22], we suggested decreasing the action space per group of drivers (e.g., 10) to train the model simpler.

B. Overall Performance Summary

Table II shows the performance of our approach (OnlineDC) against the baseline methods. We trained all algorithms using the first 10 seconds ($T = 10$). For the traditional classification baseline methods, Random Forest was the best performance (78% accuracy), while SVM was the worst performance (41% accuracy). Furthermore, MLSTM-FCN performed in second place among all algorithms with 84% accuracy. We can see that our approach (OnlineDC) outperforms for all baseline methods with at least 7% for accuracy, precision, and recall. Our approach addresses the cold-start problem with at least 8% and 9% for F1-score and Kappa, respectively.

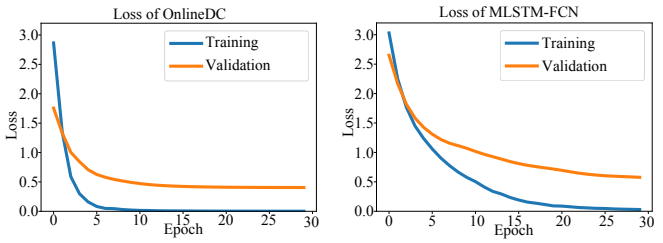

Fig. 4. The Loss of our neural network and MLSTM-FCN during training the models.

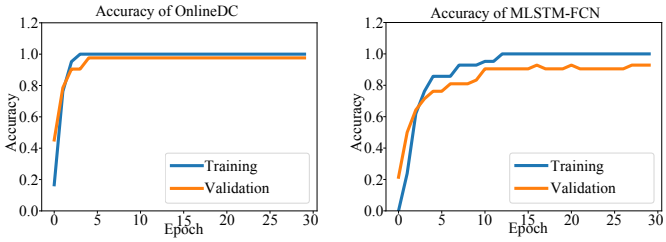

Fig. 5. The Accuracy of our neural network and MLSTM-FCN during training the models.

Fig. 4 and Fig. 5 represent the learning metrics at each iteration. We have trained each neural network once to show the learning and accuracy curves of our neural network comparable with the other neural network (MLSTM-FCN). Fig. 4 shows the loss values during training the neural networks of our approach and MLSTM-FCN. The loss is computed on the training and validation datasets. We can see that our proposed approach did better than MLSTM-FCN since the loss values were reduced before the 10^{th} iteration. Whereas, MLSTM-FCN learned slower in learning during its training to reach the optimization, as the validation's loss values start close to 0.5 at the end of 30 epochs. Furthermore, we want to show how the accuracy of our neural network model was better than the MLSTM-FCN neural network during the training of the two models as shown in Fig. 5. As the loss was reduced faster during training our model, the accuracy of our proposed neural network outperformed MLSTM-FCN.

C. The Variance of Accuracy

Fig. 6 shows the accuracy of our approach against other algorithms. We can see that OnlineDC kept the high performance in training the algorithm 10 times for the first 200 time steps ($T = 200$). Our approach has 97% accuracy except for one result (95%). Because of this, the median was 97% the same as the maximum performance. MLSTM-FCN had a high variance because it might need more training than 30 iterations as it was slower than our approach using GRU in training (Fig. 4 and Fig. 5). The results of MLSTM-FCN were between 87% and 95%. This is because GRU (used in our approach) learns faster than LSTM [40] and ResNet in OnlineDC

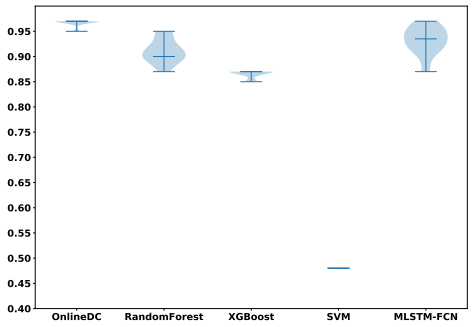

Fig. 6. Comparing the accuracy of OnlineDC with other algorithms for T=200.

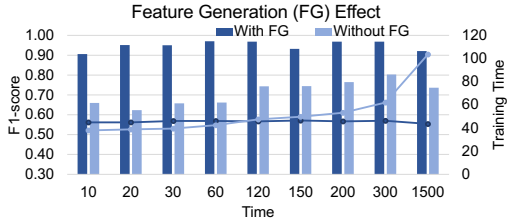

Fig. 7. Comparing the performance of OnlineDC using FG and without using FG. Bars (F1-score) and lines (Training Time).

improves performance. We can see that the variance of the results was high for the MLSTM-FCN and the Random Forest algorithm. Whereas, our approach, the XGBoost algorithm, and the SVM algorithm had accuracy with less variance. SVM suffered from chronic low performance with only 48% for all ten experiments.

D. The Effect of the FG Features

In this Section, we need to show that our Feature Generation (FG) algorithm improves the performance of our Deep Learning (MGRU-ResSE). Fig. 7 illustrates that FG is essential to foster the performance (F1-score) of our proposed approach. We can see that the generated features helped our neural network (MGRU-ResSE) to produce a classification model with high performance (i.e., F-score between 91% and 97%) for different time steps. However, training our neural network without FG built an identification model with low performance (F1-score between 62% and 80%). Additionally, the training time without FG was a little better for the first 10, 20, and 30 time steps though low performance. For the first 60, 120, 150, and 200 time steps, the training time of MGRU-ResSE with FG was so close to MGRU-ResSE without FG. After that, our neural network with FG was running faster than without FG. In general, the training time with FG was more stable and more scalable computation between 43 and 46 seconds for different time steps while the training time of our proposed neural network without FG was between 37 and 103 seconds.

E. The Importance of Using All Features

Here, we want to show how all features improve stabilizing our model. Therefore, all features (82) gave the highest accuracy for different lengths of the time series. They help

TABLE III
THE GROUPS OF THE GENERATED FEATURES

Group	Features	Total Values
Statistical without ECDF	Kurtosis, Skewness, Minimum, Maximum, Median, Standard Deviation, Variance, Mean Absolute Deviation, Root Mean Square, Interquartile Range, Histogram (10)	21
Statistical	**Statistical without ECDF**, ECDF (10), ECDF Percentile (2), ECDF Percentile Count (2)	35
Statistical and Temporal G1	**Statistical**, Absolute Energy, Centroid, Normalized Entropy, Mean Absolute Differences, Mean Differences, Median Absolute Differences	41
Statistical and Temporal G2	**Statistical and Temporal G1**, Median Differences, Negative Turning Points, Positive Turning Points, Neighbourhood Peaks, Peak to Peak, Signal Distance, Slope	48
Statistical and Temporal G3	**Statistical and Temporal G2**, Sum of Absolute Differences, Total Energy, Zero Crossing Rate	51
Statistical and Temporal	**Statistical and Temporal G3**, Sum Cubic Spline Coefficients, Sum Gradient (Squared), Sum Differences (Squared)	54
All features	**Statistical and Temporal**, Wavelet Absolute Mean (9), Wavelet Energy (9), Wavelet Entropy, Wavelet Standard Deviation (9)	82

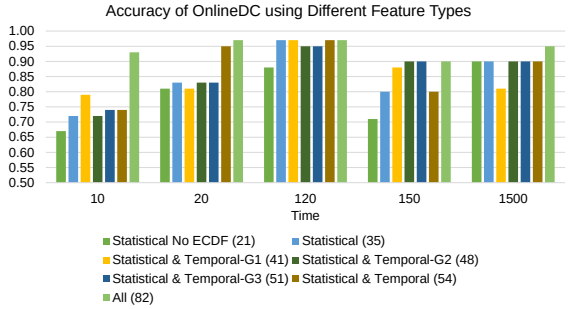

Fig. 8. Performance of OnlineDC using our neural network (GRU and ResNet with the squeeze and excite blocks) with different types of features.

the model to full fill with the driver identification requirements. All features consist of spectral, temporal, and statistical features. Fig. 8 shows the performance of our deep neural network using different feature types, while Table III shows the features in each type. In building our FG feature set, we added features incrementally to improve the performance. In some cases, we had to prune features with non-value-added such as the Autocorrelation feature that gave the same values as the Absolute Energy feature as there were no complex numbers in the signals, and Area Under the Curve feature did not improve the performance. Also, we pruned some features that decreased the performance, e.g., ECDF Slope, Wavelet (kurtosis, Skewness, Minimum, Maximum) since they increased the state space between 9 to 10 values per feature. We made sure that the output of FG would be a matrix to be fit for our neural network since GRU and Conv1D layers take inputs of 3D where the first dimension was equal to the batch size for training the model. Finally, we can see that using all

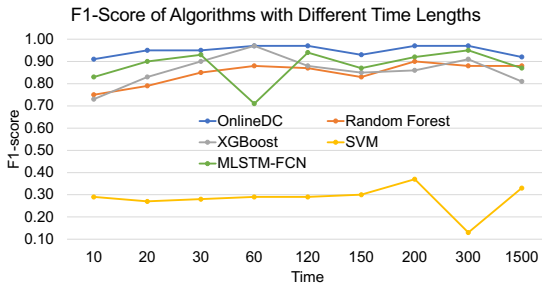

Fig. 9. Performance of OnlineDC and the baseline methods varying T.

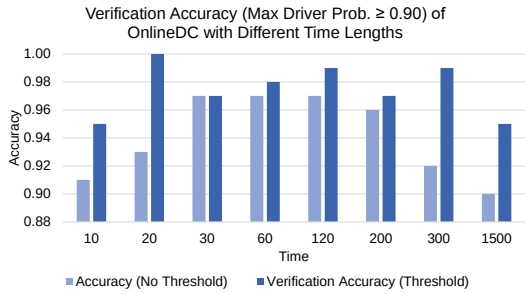

Fig. 10. Comparing accuracy without threshold and accuracy of the verification with threshold using OnlineDC varying T where the threshold is at least 90% (class probability ≥ 0.90).

features (82) gave the highest performance and produced the most stable model.

F. The Length of The Multivariate Time Series Effect

The purpose of this part was to show how our model was stable during the experiments varying the length of the original time series using our proposed feature generation algorithm and neural network. OnlineDC had a high performance (F-score 91%-97%) even though we increased the length of MTS from 10 (10 seconds) to 1500 (25 minutes) time steps. We can see that we had a larger action space (14 drivers) than [8] (2 to 5 drivers), while the performance of our approach did not decline. Hence, the results show that OnlineDC outperformed all baseline methods (Fig. 9). Finally, the classification models, generated by our approach and the baseline approaches, evaluated each driver's behavior to identify the driver in less than one second.

G. Verification Accuracy

Here, we show the automatic verification of our classification model (OnlineDC) based on the probability vector of the class labels. Previous research [4] used data similarity after classification that took a long time since it required to compare a driving behavior of a driver (test example) with all their driving behaviors. Whereas, our deep neural network provides the probability of each class label that can be used for driver verification. Fig. 10 shows the verification using 90% threshold for OnlineDC. Therefore, if the maximum probability of a class label is at least 0.90, we provide the class label (Driver ID) to the end-user in our proposed framework 3. If not, we show the unsure report with a list of three top drivers and their probability. We can see that our proposed approach provides at least 95% of accuracy. Also, the verification gives 100% accuracy for $T = 20$.

VII. Conclusion

In this paper, we used a real driving dataset with several paths. We applied both the Chi-square and the Univariate linear regression algorithms to select the most valuable signals (reducing state space). We extracted 82 features using our Feature Generation (FG) algorithm to improve performance and stabilize our driver classification model. Additionally, we designed a new MGRU-ResSE neural network architecture by utilizing GRU and ResNet with the squeeze_excite block. The conducted experiments show that our approach (OnlineDC) outperforms the baseline methods. In the future, we plan to apply our algorithm in larger real-world driving datasets and other problem domains. Additionally, we are planning to take into account the human factors in the classification model.

References

[1] J. Chen, Z. Wu, and J. Zhang, "Driver identification based on hidden feature extraction by using adaptive nonnegativity-constrained autoencoder," *Applied Soft Computing*, vol. 74, pp. 1–9, 2019.

[2] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 34–50, 2016.

[3] K. Kadiri and O. A. Adegoke, "Design of a GPS/GSM based anti-theft car tracker system." *Current Journal of Applied Science and Technology*, pp. 1–8, 2019.

[4] B. I. Kwak, J. Woo, and H. K. Kim, "Know your master: Driver profiling-based anti-theft method." in *PST*. IEEE, 2016, pp. 211–218.

[5] N. Virojboonkiate, P. Vateekul, and K. Rojviboonchai, "Driver identification using histogram and neural network from acceleration data." in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 2017, pp. 1560–1564.

[6] J. Ferreira, E. Carvalho, B. V. Ferreira, C. de Souza, Y. Suhara, A. Pentland, and G. Pessin, "Driver behavior profiling: An investigation with different smartphone sensors and machine learning," *PLoS one*, vol. 12, no. 4, p. e0174959, 2017.

[7] D. Jeong, M. Kim, K. Kim, T. Kim, J. Jin, C. Lee, and S. Lim, "Real-time driver identification using vehicular big data and deep learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 123–130.

[8] B. Kwak, M. L. Han, and H. K. Kim, "Driver identification based on wavelet transform using driving patterns," *IEEE Transactions on Industrial Informatics*, 2020.

[9] F. Martinelli, F. Mercaldo, V. Nardone, A. Orlando, and A. Santone, "Who's driving my car? a machine learning based approach to driver identification." in *ICISSP*, 2018, pp. 367–372.

[10] F. Martinelli, F. Mercaldo, and A. Santone, "Machine learning for driver detection through can bus," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.

[11] M. A. Rahim, L. Zhu, X. Li, J. Liu, Z. Zhang, Z. Qin, S. Khan, and K. Gai, "Zero-to-stable driver identification: A non-intrusive and scalable driver identification scheme," *IEEE transactions on vehicular technology*, vol. 69, no. 1, pp. 163–171, 2019.

[12] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 637–646.

[13] J.-H. Seong and D.-H. Seo, "Wi-fi fingerprint using radio map model based on mdlp and euclidean distance based on the chi squared test," *Wireless Networks*, vol. 25, no. 6, pp. 3019–3027, 2019.

[14] M. Skowron, F. Lemmerich, B. Ferwerda, and M. Schedl, "Predicting genre preferences from cultural and socio-economic factors for music retrieval," in *European Conference on Information Retrieval*. Springer, 2017, pp. 561–567.

[15] S. Ezzini, I. Berrada, and M. Ghogho, "Who is behind the wheel? driver identification and fingerprinting." *Journal of Big Data*, vol. 4, no. 1, 2019.

[16] P. H. Rettore, A. B. Campolina, A. Souza, G. Maia, L. A. Villas, and A. A. Loureiro, "Driver authentication in vanets based on intra-vehicular sensor data," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 078–00 083.

[17] I. T. Jolliffe, "Springer series in statistics," *Principal component analysis*, vol. 29, 2002.

[18] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, "Automating biomedical data science through tree-based pipeline optimization." in *Applications of Evolutionary Computation*. Springer International Publishing, 2016, pp. 123–137.

[19] R. S. Olson, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science." in *Evaluation of a tree-based pipeline optimization tool for automating data science.*, ACM. Proceedings of the Genetic and Evolutionary Computation Conference, 2016, pp. 485–492.

[20] D. Kumar, R. Singh, A. Kumar, and N. Sharma, "An adaptive method of pca for minimization of classification error using naïve bayes classifier," *Procedia Computer Science*, vol. 70, pp. 9–15, 2015.

[21] F. Tahmasbi, Y. Wang, Y. Chen, and M. Gruteser, "Poster: Your phone tells us the truth: Driver identification using smartphone on one turn." in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 2018, pp. 762–764.

[22] H. Abu-gellban, L. Nguyen, M. Moghadasi, Z. Pan, and F. Jin, "Livedi: An anti-theft model based on driving behavior," in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, 2020, pp. 67–72.

[23] S. H. Sánchez, R. F. Pozo, and L. A. H. Gómez, "Deep neural networks for driver identification using accelerometer signals from smartphones," in *International Conference on Business Information Systems*. Springer, 2019, pp. 206–220.

[24] S. H. Sanchez, R. F. Pozo, and L. A. H. Gomez, "Driver identification and verification from smartphone accelerometers using deep neural networks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[25] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification." in *Neural Networks 116*. Elsevier, 2019.

[26] H. Abu-gellban and L. Nguyen, "Cdmi: A clockwise-displacement algorithm to compute multiplicative inverse," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2020, pp. 1407–1410.

[27] S. Abujilban, L. Mrayan, S. Hamaideh, D. Alrousan, and H. Abu-gellban, "Attitudes toward corona vaccine and intention among university students," to appear in 2021 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2021.

[28] L. St, S. Wold *et al.*, "Analysis of variance (anova)," *Chemometrics and intelligent laboratory systems*, vol. 6, no. 4, pp. 259–272, 1989.

[29] A. Amiri-Simkooei, M. Hosseini-Asl, J. Asgari, and F. Zangeneh-Nejad, "Offset detection in gps position time series using multivariate analysis," *GPS Solutions*, vol. 23, no. 1, p. 13, 2019.

[30] Z. Pan, H. L. Nguyen, H. Abu-gellban, and Y. Zhang, "Google trends analysis of covid-19 pandemic," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 3438–3446.

[31] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, 1980.

[32] Ç. Kocaman and M. Özdemir, "Comparison of statistical methods and wavelet energy coefficients for determining two common pq disturbances: Sag and swell," in *2009 International Conference on Electrical and Electronics Engineering-ELECO 2009*. IEEE, 2009, pp. I–80.

[33] B. Yan, A. Miyamoto, and E. Brühwiler, "Wavelet transform-based modal parameter identification considering uncertainty," *Journal of Sound and Vibration*, vol. 291, no. 1-2, pp. 285–301, 2006.

[34] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.

[35] S. Raschka, "Mlxtend: providing machine learning and data science utilities and extensions to python's scientific computing stack," *Journal of open source software*, vol. 3, no. 24, p. 638, 2018.

[36] H. Abu-gellban, L. Nguyen, and F. Jin, "Gfdlecg: Pac classification for ecg signals using gradient features and deep learning," in *Advances in Data Science and Information Engineering*. Springer, 2021, pp. 369–382.

[37] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids." *Mathematics of computation*, vol. 41, no. 184, 1988.

[38] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning." in *15th HotNets*. ACM, 2016, pp. 50–56.

[39] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation." in *arXiv preprint arXiv*, 2014.

[40] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example," in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*. IEEE, 2020, pp. 98–101.

[41] L. H. Nguyen, Z. Pan, O. Openiyi, H. Abu-gellban, M. Moghadasi, and F. Jin, "Self-boosted time-series forecasting with multi-task and multi-view learning." in *arXiv preprint arXiv:1909.08181*, 2019.

[42] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling." in *arXiv preprint arXiv:1412.3555*, 2014.

[43] M. N. Moghadasi, Y. Zhuang, and H. Gellban, "Robo: A counselor chatbot for opioid addicted patients," in *2020 2nd Symposium on Signal Processing Systems*, 2020, pp. 91–95.

[44] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation." in *CVPR*. IEEE, 2015, pp. 3431–3440.

[45] R. L. Kumar, J. Kakarla, B. V. Isunuri, and M. Singh, "Multi-class brain tumor classification using residual network and global average pooling," *Multimedia Tools and Applications*, vol. 80, no. 9, pp. 13 429–13 438, 2021.

[46] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks." in *CVPR*, 2015, pp. 7132–7141.

[47] V. Kougia, J. Pavlopoulos, and I. Androutsopoulos, "Aueb nlp group at imageclefmed caption 2019." in *CLEF (Working Notes)*, 2019.

[48] F. Karim, S. Majumdar, H. Darabi, and C. Chen, "LSTM fully convolutional networks for time series classification." in *IEEE Access 6*. IEEE, 2017.

[49] D. Hallac, A. Sharang, R. Stahlmann, A. Lamprecht, M. Huber, M. Roehder, and J. Leskovec, "Driver identification using automobile sensor data from a single turn." in *In 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 953–958.

[50] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system." in *22nd ACM SIGKDD*. ACM, 2016, pp. 785–794.

[51] J. Zhou, E. Li, M. Wang, X. Chen, X. Shi, and L. Jiang, "Feasibility of stochastic gradient boosting approach for evaluating seismic liquefaction potential based on SPT and CPT case histories." in *Journal of Performance of Constructed Facilities 33, no. 3*, 2019.

[52] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud." in *BigComp*. IEEE, 2018, pp. 251–256.

[53] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications." in *In 2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 3854–3861.

[54] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline." in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

[55] L. H. Nguyen, S. Jiang, H. Abu-Gellban, H. Du, and F. Jin, "Nipred: Need predictor for hurricane disaster relief," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 190–193.

[56] H. Abu-gellban, "A survey of real-time social-based traffic detection," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2020, pp. 1–6.

[57] H. Du, L. Nguyen, Z. Yang, H. Abu-Gellban, X. Zhou, W. Xing, G. Cao, and F. Jin, "Twitter vs news: Concern analysis of the 2018 california wildfire event," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2019, pp. 207–212.

[58] P. H. L. Rettore, A. B. Campolina, L. A. Villas, and A. A. F. Loureiro, "A method of eco-driving based on intra-vehicular sensor data," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 1122–1127.