IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.20xx.DOI

Driving Maneuver Classification Using Domain Specific Knowledge and Transfer Learning

SUPRIYA SARKER[1], MD. MOKAMMEL HAQUE[1], AND M. ALI AKBER DEWAN.[2], (Member, IEEE)

[1]Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh
[2]School of Computing and Information Systems, Faculty of Science and Technology, Athabasca University, Athabasca, AB T9S 3A3, Canada

Corresponding author: Md. Mokammel Haque (e-mail: mokammel@cuet.ac.bd).

ABSTRACT With the increasing number of vehicles, the usage of technology has also been increased in the transportation system. Although automobile companies are using advanced technologies to develop high performing transports, traffic safety still remains to be a concerning issue. Drivers' driving behavior is considered as one of the key factors of the traffic safety, which could be monitored from their individual driving maneuvers. In this paper, we present a supervised learning model and a semi-supervised transfer learning model for the classification of driving maneuvers from the sensor fusion time series data. The semi-supervised model consists of an unsupervised long-short term memory (LSTM) autoencoder and a supervised LSTM classifier. The supervised model consists of a supervised LSTM model. Because of using LSTM, both of the models can analyze time-series data. In the semi-supervised model, the LSTM encoder learns from unlabeled data as a compressed low dimensional feature vector, which then transfers the learning to the supervised LSTM classifier to classify the driving maneuvers. With the proposed models, we use domain specific knowledge data of the driving environment, such as data changing rules of various driving maneuvers as well as the temporal features over time. We use class functions for seven driving maneuver types and convert those into binary feature vector to use with the LSTM models. We present a comparative analysis of the per class accuracy of the proposed semi-supervised and supervised models with and without using domain-specific knowledge, where the models with the domain specific knowledge outperform. Our proposed semi-supervised and supervised models are compared with the other existing approaches, where our models trained with the domain specific knowledge provide better performance. We also compare the per class accuracy for both the supervised and semi-supervised models, where all the maneuver class accuracy for supervised model is above 98% and semi-supervised model is above 95%. Although the supervised model outperforms the semi-supervised model, semi-supervised model would be more beneficial in applications where the labeled driving maneuvers data is hard to capture or insufficient.

INDEX TERMS Driving maneuver classification, domain specific knowledge, LSTM autoencoder, semi-supervised learning, transfer learning.

I. INTRODUCTION

TRANSPORTATION system has greatly influenced by the industrial revolution. With the expanding number of vehicles, the concern of traffic safety is growing concurrently. Though massive endeavors have been taken over the decades to ensure road safety by adopting new technologies, traffic safety is still a concerning issue [1]– [4]. Drivers' driving behavior has a great impact on accidents. Recent studies have shown that the knowledge of predictive driving assistant systems about the intention of the driver can be utilized to notify about dangerous driving and alleviate traffic mishap [5]– [11].

The two general ways of collecting information regarding moving vehicle are Controller Area Network (CAN) [12] bus and Micro Electro Mechanical System (MEMS) [13]. Through CAN bus one vehicle can communicate to other vehicles using microcontrollers and other devices without a host computer and bear all the required information to recognize the state of the vehicle. CAN bus information can be access using On-Board Diagnostic (OBD) port. However, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

capability of accessing the vehicle information through OBD port relies on the expertise of private protocol of the vehicle [14]. Recently, MEMS is getting much popular because of its compact size, lightweight, energy efficiency. MEMS can be mounted in the subject vehicle to integrate several sensors e.g., accelerometer, gyroscope, GPS, and so on [15].

Various approaches have been adopted by utilizing inertial signals from various sensors such as accelerometer, gyroscope, GPS, video to monitor driving maneuvers [16], [17]. However, excessive input worsens the performance of the system and increases computational cost [14]. Nonetheless, it is proven in previous research that inertial sensors are capable of recognizing particular driving maneuvers [14]. With the spread of smartphones, few researchers successfully exploited smartphone in sensor platform [16], [18], [19]. They mounted smartphones with sufficient amount of sensors in the subject vehicle and the collected data was stored inside the smartphone. However, in a real-life situation, this setup may be interrupted because the drivers may use smartphone for communication and navigation [14]. To overcome this problem, some researchers utilized previously collected sensor fusion dataset [20]–[22]. Researchers proposed various fuzzy inference, machine learning, deep learning techniques to classify driving behavior from inertial sensors data [20]. However, they ignored the domain-specific knowledge of the driving environment. The knowledge of the environment where the data belong to is referred as domain-specific knowledge. During movement, a vehicle can be considered as a rigid body and the change rule of the data collected by the sensors during various driving maneuvers can be explained by the theory of rigid body kinematics [14]. The change rule reveals secret patterns that can be considered the domain specific knowledge of driving data. The domain-specific knowledge of driving data is discussed elaborately in Section II. The negligence domain-specific knowledge restricts further optimization and improvements. Therefore, it is necessary to examine the impact of domain-specific knowledge in the performance of driving maneuvers classification.

Moreover, most of the Machine Learning (ML) or Deep Learning (DL) techniques were based on supervised learning by utilizing the labeled driving dataset. The data that contains maneuver class information is refereed to as labeled data. On the other hand, if the data does not have maneuver class information can be referred to as unlabeled data. However, labeling is an expensive and time-consuming task. That is why in most of cases, a small amount of data is being labeled and a large amount of data remains unlabeled. Therefore, a vast amount of unlabeled data which may contain potential patterns, cannot participate in the classification task. Therefore, a system needs to develop to utilized the complete dataset (both labeled and unlabeled) for classification of driving maneuvers.

Prior to classification, learning the data without label i.e., unlabeled data is vital and challenging. A common phenomenon demonstrated in several previous research is the extraction of hand-engineered features prior to applying any classification algorithm and believed that local, salient patterns of time series data had been being detected and being extracted [23]. But, unstable, irregular, and vigorously emerging behavior of the sensor data demand ingenious machine learning framework to apprehend temporal relationships in the time series data. Hence, the performance of the time series classification extremely reliant on the expertise of domain variable selection of domain experts [23]. Certain types of problems that require a classifier can be benefited by the transformations of the features because the performance and precision of the classifier model heavily dependent on the quality of the learned features. Leading-edge advancements of deep learning networks provide an advantage by integrating the latent representation within the classifiers as a form of Autoencoder [24]. Autoencoder can be a possible solution to many problems because of its structure as well as objective function which can be modeled as a transformation of the feature space [25]. The compressed representation of the time-series data obtained from Autoencoder can contribute as input to another supervised trained classifier.

Therefore, the research question addressing in this paper is (1) "How to develop a neural network model to classify driving maneuvers from sensor fusion time series labeled data by training with domain-specific knowledge?" (2) "How to develop a neural network model to classify the driving maneuvers from sensor fusion time series unlabeled data?" To address these research questions in this work, we develop (i) an *LSTM Network* for classification of driving maneuvers from sensor fusion time series labeled dataset incorporating domain-specific knowledge of moving vehicle; (ii) an *LSTM Autoencoder* for the latent representation of unlabeled dataset and transfer learning to proposed supervised model for classification.

The contributions in this paper can be summarized as follows:

- We develop few class functions from domain specific knowledge and convert them into binary domain specific feature vectors to train the proposed model.
- We propose an LSTM network for supervised classification of driving maneuvers with improved accuracy compared to other related works. For this purpose, we train the model with binary domain specific feature vector along with other temporal features.
- We propose an LSTM autoencoder for the compressed representation learning so that latent features learning can be transfer to train the proposed supervised LSTM model.

This paper states the importance of the classification of driving maneuvers in Section I. II. Literature review are discussed in Section II. The theoretical framework is explained in Section III. The methodology of the proposed deep supervised and unsupervised learning techniques has been discussed in Section IV including preparation of dataset and feature set extraction. Section V exploits and analyzes the results including hyperparameter optimization. Section VI

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

concludes the paper with future work.

II. LITERATURE REVIEW

In this section, we will discuss the previous related works for driving maneuvers recognition. From the literature review, it is noticeable that several techniques have been proposed and developed to recognize driving maneuvers. In the following subsections, we discuss about the classical approaches, machine learning approaches, and deep learning approaches.

A. CLASSICAL APPROACHES

In this category, we discuss the approaches that included fuzzy inference rules, dynamic time warping, motif discovery techniques in time-series data. Saiprasert *et al.* [16] proposed a profiling algorithm that categorized drivers' behavior using sensory data collected from smartphones in a bidirectional vehicle-to-infrastructure environment. Drivers were categorized according to four different driving events namely acceleration, braking, turning, and lane changing with the most risk to cause an accident during their journey. To measure categories particular Safety Index (SI) was used for all events comprised of sudden acceleration, weights of acceleration. Driver's profile aimed to create an individual profile of each driver considering safe and aggressive driving style. However, the safety index was generated by combining the index of all the sudden events. Hence, it is unable to measure the driving risk associated with a particular maneuver. The profiling algorithm calculated complex road geometry (i.e., curves and turns) by the higher value of sudden turn, lane change than sudden acceleration, braking while a machine learning approach can do weighing more efficiently. Moreover, the categories were created based on accelerometer data which is not completely able to determine the angular change of moving vehicle. Johnson *et al.* [18] proposed a system that used smartphone-based sensor-fusion data such as accelerometer, gyroscope, magnetometer, GPS, and video to classify the type of driving maneuver and style of driving as aggressive or nonaggressive. The authors used x-axis of gyroscope, y-axis of accelerometer, z-axis of device Euler angle rotation for detecting turning and y-axis of gyroscope, z-axis of acceleration for longitudinal movements with Dynamic Time Warping (DTW) [26]. However, non-aggressive lane change has not been possible to detected by the system during the experiment because of lack of force or rotation on the device to differentiate from noise. Only normal and turn events has been detected by the DTW [26] algorithm. Castignani *et al.* [19] developed a Fuzzy Inference [27] based web-based a mobile tool to evaluate and score the overall driving behavior in terms of acceleration, over speed, and steering/bearing rate when turning at the intersections by utilizing accelerometer, magnetometer, gravity sensor, and GPS data. All the sensor data stored in a remote database and based on a few predefined threshold values of each possible event the system provided a score of all the driving events. The authors combined 12 input variables and the profiling algorithm used 18 fuzzy rules and outputted fuzzy sets consisting of normal, moderate, and aggressive driving styles. However, they suspected that there might be other rules those were not considered in the work. Schwarz *et al.* [28] categorized driving patterns from time- series naturalistic driving data using Symbolic Aggregate Approximation (SAX) [29] and Matrix profile method [30]. The data included maneuvers like turning, stopping at intersections, parking, and leaving parking spaces . The authors performed a speed and rotational transformation of acceleration signal rather than DTW and to discover motif in time-series acceleration was codded with six alphabets (a-f) where each letter expressed aggregated mean across five meters. However, the work has not mentioned that how different maneuvers were differentiate among each other through motifs.

B. MACHINE LEARNING APPROACHES

In this category, we discuss the related machine learning approaches that have been proposed for the classification of driving maneuvers. Van Ly *et al.* [31] applied Support Vector Machine [32] and K-mean clustering [33] to build an individual driver profile providing proper feedback to reduce the number of dangerous car driving maneuvers. The work best recognized braking while acceleration is the less distinctive feature between two drivers and the provided best accuracy was 60% and 65% for K-means clustering and SVM, respectively. Cervantes-Villanueva *et al.* [34] proposed a speed-based breakout detection agent for detecting sudden speed variation and maneuver detection agent for classifying four driving states which are stopped, driving, parking, and parked. They applied Random Forests (RF) [35], Support Vector Machines (SVM) [32], and fuzzy rule-based classifiers as maneuver detection agent. For this purpose, the authors utilized accelerometer data that was collected by smartphone. They found that RF provided best results in the two-level agents or classifiers. In addition, the detection of change of speed to activate the classifier reduce the overall accuracy of the classifier but improved the computational cost. Besides, they did not consider the kinematic states of vehicles which involves angular motion. Ferreira *et al.* [20] evaluated the quantitative performance of Multi-Layer Perceptron (MLP), Support Vector Machines, Random Forrest, Bayesian Networks [36] in classification of driving maneuvers from smartphone sensors. Also, the author did a comparative analysis of multiple combinations of the supervised machine learning algorithms to classify seven driving maneuvers class. The authors also analyzed the performance using multiple combinations of sensors with various axis such as x, y, and z-axis of accelerometer, gyroscope, and magnetometer for varied size sliding window. The authors found that bigger sliding window improved the accuracy. As well as, accelerometer, gyroscope with most axes provided better performance. Among machine learning algorithms, RF provided best performance. However, access sensor axes increases the computational cost of the classifiers [14]. Besides, as the author applied supervised techniques, a small amount of labeled dataset has been used for classification and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

potential unlabeled time-series datasets were left unused.

Wang *et al.* [37] proposed a semi-supervised support vector machine (S3VM) approach to classify aggressive and normal longitudinal driving style using a small amount of labeled data. In order to labeled few data points, they applied k-means clustering method and then few clusters were labeled using rule based approach. To solve the optimization problem, they introduced a specific differentiable surrogate of a loss function and to assign label quasi-Newton algorithm was used with semi-supervised SVM. For feature selection they had considered vehicle speed, throttle opening with certain threshold range. All the driving data was collected from a driving simulator; hence, real traffic scenarios were not considered. They provided a comparison of their semi-supervised SVM with supervised SVM and found that proposed semi supervised SVM showed 10% increase accuracy.

C. DEEP LEARNING APPROACHES

In this category, we discuss the related approaches that used deep learning techniques for driving maneuvers classification. Carvalho *et al.* [21] examined the performance of three deep neural network which are Recurrent Neural Network (RNN) [38], Long Short Term Memory (LSTM) [39], and Gated Recurrent Unit (GRU) [40] to classify seven driving maneuvers from accelerometer data collected by smartphone. However, only accelerometer data cannot present all the changes during vehicle movement [14], [20]. The performance of the model for classifying each class was not mentioned in the work. Alvarez-Coello *et al.* [22] proposed and split the supervised multi-class driving maneuver classification problem into two parts. The authors developed a binary classifier by applying RF in order to classify aggressive and non-aggressive driving events and the result was transferred to the RNN model to recognize the type of maneuver. Among the variant of RNN, LSTM performed better than GRU for most of the combination in their experiment. However, the authors mentioned that the used dataset was labeled manually and has a possibility of bias with the labelers' perception. Sarker *et al.* [41] established a few hypotheses based on domain-specific knowledge of moving vehicles which describe the kinematic state change of moving vehicle [14] and proposed an LSTM network for the multi-class classification of driving maneuvers from sensor fusion time series data. In their proposed deep learning model, the authors trained the classifier model by extracting features and domain-specific knowledge of vehicle kinematics which enhance the performance, precision, and time efficiency. However, like other previous work, they utilized labeled datasets for supervised techniques. We consider this approach as our primary reference extend the work in semi-supervised manner.

Another Semi-Supervised deep learning approach adopted by [42]. Mammeri *et. al* [42] proposed a deep semi-supervised approach utilizing manually few labeled driving data considering three CAN bus parameters i.e., velocity, acceleration, and steering wheel angle. They proposed a coarse thresholding strategy to label data from video, refined the created labels to reduce error and trained a simple Convolutional Neural Network (CNN) to classify ten driving maneuvers subclasses which were stop, move, acceleration, deceleration, constant speed, left/right turning, left/right curving and constant direction under three groups (motion, velocity, turning). After manual refinement they computed 87.3%, 78.4% and 76.9% for motion, velocity, and turning. After training CNN model with the complete dataset they computed overall accuracy 93.48% with 165 out of 4705 samples for training where 66 sample was labeled. However, they have not discussed subclass accuracy. However, they have trained the network with all samples and have not mention from where the test data has come. If they used a fragment of the dataset which means that the network was pre-trained with the test data also and its easy to predict the maneuver class of the test set for the model. Hence, definitely the accuracy will be high.

Prior researchers who implemented fuzzy rule based system, measured threshold of various states of moving vehicle and depending on these values, they proposed equations to calculated index of drivers' behavior. This index value classify the drivers' behaviour as aggressive, moderate or non-aggressive. The limitations of this system is building fuzzy rule for every circumstances is quite difficult. Researcher who applied machine learning and deep learning approaches, generally focused on supervised techniques using labeled sensor fusion data. Obtaining labels data for every driving maneuvers is difficult and expensive. As the data is being labeled by human manually, the perfection of the maneuver class label is heavily depends on the maneuver perception of a labeler. Hence, there are possibilities of bias [22]. Besides, most cases, a small amount of labeled dataset can be found which further participate to train models. In addition, none of the earlier researcher noticed the relationships of kinematic states and maneuvers except for [14]. In this research, we focus on the relationship of particular maneuvers with sensor data changing nature and extract these features in feature extraction process. In order to learn the complete dataset (both labeled and unlabeled) and train the deep model with these, we try to develop a semi-supervised mechanism by transferring the latent space representation of dataset to our proposed supervised deep model.

III. THEORETICAL BACKGROUND

In this section we discuss the theoretical background of moving vehicle and fundamentals of deep learning framework. Besides, the significance of domain knowledge in machine learning approaches is also illustrated. The Table 1 interprets all the important symbols used throughout the paper.

A. DOMAIN SPECIFIC KNOWLEDGE OF MOVING VEHICLE

It is very important to understand the data within the context of the problem that we are trying to solve before modeling. Domain knowledge can often guide us to understand the pre-processing, finding significant features and hence, improve

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

TABLE 1. Interpretation of symbols

Symbol	Interpretation
$\overrightarrow{a}$	acceleration in a direction
$\overrightarrow{w}$	angular velocity in a direction
$\overrightarrow{\mathrm{v_t}}$	velocity in time t
a_x, a_y, a_z	accelerometer data in x, y, z axis, respectively
w_x, w_y, w_z	gyroscope data in x, y, z axis, respectively
$a_{x_{slp}}$, $a_{y_{slp}}$	slope of accelerometer in x, y axis, respectively
$w_{z_{slp}}$	slope of gyroscope in z axis
$a_{x_{eng}}$, $\mathrm{a}_{y_{eng}}$	energy of accelerometer in x, y axis, respectively
$w_{z_{eng}}$	energy of z axis of gyroscope
$f(t_{acc})$, $f(t_{br})$, $f(t_{llc})$ $f(t_{rlc})$, $f(t_{lt})$, $f(t_{rt})$ $f(t_{non})$	Class function of aggressive acceleration, braking, LLC, RLC, LT, RT and non-aggressive maneuvers, respectively
f_S, f_D	Statistical, Domain-Specific feature set, respectively
h_t, m_t	hidden vector, memory vector, respectively
i_g, f_g, c_g, o_g	input, forget, update, output gate of LSTM network, respectively
x_t, $\tilde{x}_t$	Input, reconstructed data of Autoencoder, respectively
l_1, l_2, l_3	Input Layer of Autoencoder
l_5, l_6, l_7	Reconstructon layer of Autoencoder
u, v, z	Output unit of Autoencoder layer
l_4, l_8	RepeatVector, TimeDistributed layer of Autoencoder, respectively

the precision and accuracy of the model. Domain knowledge is also essential to deal with a specific problem and the modeling and evaluation process can be vary depending on it. Instead of labeling, neural networks can be trained to capture mathematical and logical relationships [43]. Stewart and Ermon in [43] supervised a neural network model by the physics of free-falling objects rather than training directly on labels and compared the model trained on labels and domain knowledge-based physical laws. Therefore, before building a classification model we need to understand the domain-specific knowledge of the real traffic scenarios.

In our target domain of moving vehicle, there are some changes in kinematic states such as acceleration, deceleration, angular velocity, etc. of the subject vehicle while performing particular movements that is maneuvers. These change of states follows a few specific rules that can be described by the theory of rigid body kinematics and have been discussed by Wu *et al.* in [14]. When a vehicle move forward it produces some longitudinal displacement that is acceleration and deceleration which can be defined by the first kinematic formula. In t, the acceleration, $\overrightarrow{a}$ can be defined by (1). While performing a lateral movement, the vehicle produces lateral displacement and angular velocity. During turning around a radius, r, the angular velocity, $\overrightarrow{w}$ of a vehicle can be defined by (2).

$$\overrightarrow{a} = \frac{\overrightarrow{v_t} - \overrightarrow{v_0}}{t} \tag{1}$$

$$\overrightarrow{w} = \frac{\overrightarrow{\mathrm{v_t}}}{t} \tag{2}$$

In (1), $\overrightarrow{v_0}$ is the velocity in 0 second and $\overrightarrow{v_t}$ is the velocity in t second. When $\overrightarrow{v_t} > \overrightarrow{v_0}$, then $\overrightarrow{a} > 0$; hence, at the beginning of acceleration, the time series data increases from zero to a greater value and at the end, it decreases to a lower value. The pattern during acceleration is shown in the Fig. 1. On the contrary, while $\overrightarrow{v_t} < \overrightarrow{v_0}$ then $\overrightarrow{a} < 0$. So, the time series data shows opposite pattern in the negative x axis, referred as braking and shown in the Fig. 1. From (2), $\overrightarrow{w}$ proportional to $\overrightarrow{v_t}$ in time t, therefore, change of angular velocity is persistent to acceleration, particularly in this context, lateral acceleration.

This change of states can be captured as continuous time-series data through various sensors such as accelerometer, gyroscope, magnetometer, etc. Each of these sensors has three axes in x, y, and z-direction. In this paper, we denote accelerometer data by a_x, a_y, a_z and gyroscope data by w_x, w_y, w_z in x, y, z dimension, respectively. Since a_x, a_y and w_z data can recognize the common driving maneuvers [14], we consider these data only. It is illustrated in the Fig. 1 that when vehicle moves in longitudinal distance that is, at the time of acceleration and braking a_x shows meaningful change in pattern. Similarly, because of lateral displacement and angular velocity in similar direction a_y and w_z shows significant change in pattern during left, right turn and left, right lane change.

By plotting driving data, Wu *et al.* [14] and Sarker *et al.* [41] found that sensor fusion time-series driving data follows some threshold values for a particular axis during a particular maneuver. The threshold for each axis is illustrated in Fig. 1. During non-aggressive acceleration and braking, a_x data ranges from zero to 2 and -2, respectively. a_x data is not much meaningful for recognizing Left Lane Change (LLC) and Right Lane Change (RLC). During Left Turn (LT) and Right Turn (RT) a_y rise from 0 to 1.5 and reaches to negative axis in the same amount. Besides, w_z follows the same pattern but ranges 0 to 2 for LT and 0 to -2 for RT. Along with sensor fusion data, statistical features such as mean, local maxima, minima, variance, standard deviation, slope, energy follow a specific threshold for a particular maneuver. We found that among the statistical features, slope, and energy capable of making some important differences in decision making of machine learning model. So, we convert few domain specific binary features from class functions. The class functions are discussed in Section III-B.

B. CLASS FUNCTIONS FROM DOMAIN SPECIFIC KNOWLEDGE

Sarker *et al.* [41] investigated and found that these changing pattern and their corresponding threshold has significance during feature set extraction of driving maneuver classification. They developed seven class hypotheses based on observation of labeled data patterns and provided some threshold for each maneuver class which can be considered as class

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

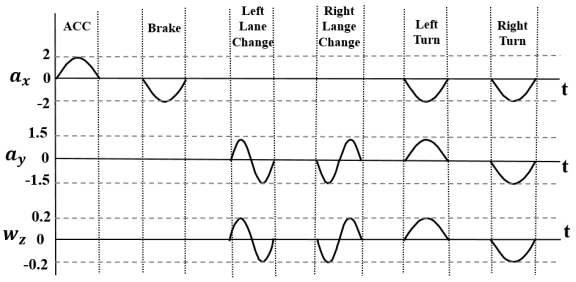

FIGURE 1. Illustration of time series data pattern of a_x, a_y and w_z during acceleration, braking, left lane change, right lane change, left turn and right turn in time t. The vertical axis shows the safe threshold value of a_x, a_y and w_z, particularly.

functions of corresponding maneuver class. The functions are as follows:

- Class 1: Aggressive Acceleration

$$f(t_{acc}) = \{(a_x > 2) \vee (a_{x_{slp}} > 5); \tag{3}$$

- Class 2: Aggressive Brake

$$f(t_{br}) = \{(a_x < -2) \vee (a_{x_{slp}} < -5); \tag{4}$$

- Class 3: Aggressive Left Lane Change

$$f(t_{llc}) = \begin{cases} (a_y > 2) \vee (a_{y_{slp}} > 5); \\ (w_z < -0.2) \vee (w_{z_{slp}} < -0.6); \end{cases} \tag{5}$$

- Class 4: Aggressive Right Lane Change

$$f(t_{rlc}) = \begin{cases} (a_y < -2) \vee (a_{y_{slp}} < -5); \\ (w_z > 0.2) \vee (w_{z_{slp}} > 0.6); \end{cases} \tag{6}$$

- Class 5: Aggressive Left Turn

$$f(t_{lt}) = \begin{cases} (a_y > 1.5) \vee (a_{y_{slp}} > 5) \\ \vee (a_{y_{eng}} > 0.3); \\ (w_z > 0.2) \vee (w_{z_{slp}} > 0.6) \\ \vee (w_{z_{eng}} > 0.3); \end{cases} \tag{7}$$

- Class 6: Aggressive Right Turn

$$f(t_{rt}) = \begin{cases} (a_y < -1.5) \vee (a_{y_{slp}} < -5) \\ \vee (a_{y_{eng}} > 0.3); \\ (w_z < -0.2) \vee (w_{z_{slp}} < -0.6) \\ \vee (a_{y_{eng}} > 0.3); \end{cases} \tag{8}$$

- Class 7: Non Aggressive

$$f(t_{non}) = t_i \notin \{f(t_{acc}), f(t_{br}), f(t_{llc}), f(t_{rlc}), f(t_{lt}), f(t_{rt})\} \tag{9}$$

Here, the hypotheses are being considered as function of time, t defined by (3) to (9), respectively, where $f(t_{acc})$, $f(t_{br})$, $f(t_{llc})$, $f(t_{rlc})$, $f(t_{lt})$, $f(t_{rt})$ and $f(t_{non})$ is function of time series that represent event of aggressive acceleration, braking, left lane change, right lane change, left turn, right turn, and non-aggressive maneuver, respectively. $a_{x_{slp}}$, $a_{y_{slp}}$ and $w_{z_{slp}}$ represents slope of a_x, a_y, w_z, respectively and $a_{x_{eng}}$, $a_{y_{eng}}$ and $w_{z_{eng}}$ represents energy of a_x, a_y, w_z, respectively. Eq. (3) implies that if a_x data is greater than 2 or slope of a_x data is greater than 5 the data belongs to $f(t_{acc})$ i.e., Aggressive Acceleration class. Similarly, (4) implies that if a_x data is less than -2 or slope of a_x data is less than -5 the data belongs to $f(t_{br})$ i.e., Aggressive Brake class. If a_y data is greater than 1.5 or $a_{y_{slp}}$ greater than 5, then the data belong to Aggressive Left Lane Change class function, $f(t_{llc})$. Besides, if w_z is less than -0.2 or $w_{z_{slp}}$ is less than -0.6 the data belong to $f(t_{llc})$. Eq. (6) denotes that if a_y data is less than 1.5 or $a_{y_{slp}}$ less than 5, then the data belong to Aggressive Right Lane Change class function, $f(t_{rlc})$. Also, if w_z is greater than 0.2 or $w_{z_{slp}}$ is greater than 0.6 the data belong to $f(t_{rlc})$. a_y data which is greater than 2 or $a_{y_{slp}}$ is greater than 5 or $a_{y_{eng}}$ is greater than 0.3 then data belong to Aggressive Left Turn class function, $f(t_{lt})$. Moreover, if w_z data greater than 0.2 or $w_{z_{slp}}$ is greater than 0.6 or $w_{z_{eng}}$ is greater than 0.3 the data belong to $f(t_{lt})$. a_y data which is less than -2 or $a_{y_{slp}}$ is less than -5 or $a_{y_{eng}}$ is greater than 0.3 then data belong to Aggressive Right Turn class function, $f(t_{rt})$. Also, if w_z data is less than -0.2 or $w_{z_{slp}}$ is greater than -0.6 or $w_{z_{eng}}$ is greater than 0.3 the data belong to $f(t_{rt})$. Time series data that do not fall into class 1 to 6 will be non-aggressive and belong to Non Aggressive class function, $f(t_{non})$.

Since the above functions contain domain-specific knowledge of moving vehicle maneuvers we convert these class functions into binary features vector. A fragment of the domain-specific knowledge vector is illustrated in Table. 2 and is discussed in the Section IV-B.

C. DEEP LEARNING FRAMEWORK

Deep learning is a sub-field of broader machine learning family involved with algorithms inspired by artificial neural networks. A neural network with at least two-layer referred to as a deep neural network. It creates a map of neurons and assigns weights for connections. Each connection provides output by multiplying weights to input and adjust weight vectors until the model able to accurately determine a pattern. The computation of a deep neural model involves complex data processing with convoluted mathematical modeling. Deep architecture has many variants suitable for a specific domain. In our work, we develop an LSTM model for driving maneuvers classification from time-series data.

D. DEEP UNSUPERVISED REPRESENTATION LEARNING FRAMEWORK

Neural Networks have the idiosyncrasy of being organized as required for solving a problem. A different form of the neural network is AutoEncoder (AE) which can be used as a standalone feature learner [25]. The fundamental principle of AE is achieving a low dimensional latent representation by optimizing a local unsupervised criterion defined by the loss function, each layer is being trained at once to produce a meaningful higher-level compressed representation of original input, eventually to enhance the generalized representation [44]. In particular, AE is a neural network designed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

in such a way that a bottleneck is imposed in the network which compels a compressed knowledge representation of the original input.

A simple AE consists of three components: Encoder, Code or Latent Space Representation, and Decoder. The encoder compresses the input into a latent space representation in a reduced dimension. Code is the part of the network that represents the compressed input that is fed to the decoder. The decoder layer decodes the encoded input from the latent space representation. The process of training of AE is analogous to a feedforward neural network through backpropagation [45].

Since the training process of AE does not require explicit labels to train on in order to generate a model for the data, it can be utilized to learn unlabeled data. AE is only able to compress meaningful data similar to what it has been trained on. It is lossy by its nature which means the output of the AE will not be exactly the same as the input. It will be close but degraded reconstruction. The target of training of an AE is to minimize the reconstruction error by optimizing the hyperparameters of the network [45].

In this work, we develop an LSTM Autoencoder to learn unlabeled time-series datasets and build a compressed feature vector which in turn fed into our supervised model for classification of driving maneuvers.

IV. METHODOLOGY

The main objectives of our work are two-fold. First of all, we develop a supervised classifier in order to solve the multiclass driving maneuver classification problem using a deep learning approach (i.e., LSTM). This classifier is being trained with labels i.e., class information of time series sensor fusion data. Afterward, we develop an unsupervised LSTM AE model to learn latent space representation of the complete dataset without the class information. Finally, the latent space representation of input data is transferred through the supervised LSTM classifier model to predict the classes of the maneuver.

A. PREPARATION OF DATASET

Let, the set of time series data from the x-axis of the accelerometer sensor is S_{a_x}= $\{a_{x_1}, a_{x_2}, \ldots, a_{x_n}\}$, the set of time series data from the y-axis of the accelerometer sensor is S_{a_y}= $\{a_{y_1}, a_{y_2}, \ldots, a_{y_n}\}$ and the set of time series data from the z-axis of the gyroscope sensor is S_{w_z}= $\{w_{z_1}, w_{z_2}, \ldots, w_{z_n}\}$, where n is the number of time series. The set of class, C = $\{\mathrm{C}_{acc}, \mathrm{C}_{br}, \mathrm{C}_{llc}, \mathrm{C}_{rlc}, \mathrm{C}_{lt}, \mathrm{C}_{rt}, \mathrm{C}_{non}\}$ where C_{acc}, C_{br}, C_{llc}, C_{rlc}, C_{lt}, C_{rt} and C_{non} represent class of aggressive acceleration, braking, LLC, RLC, LT, RT and non-aggressive maneuver, respectively. Statistical feature set, f_S = $\{mean_{(a_x, a_y, w_z)}, var_{(a_x, a_y, w_z)}, std_{(a_x, a_y, w_z)}, slp_{(a_x, a_y, w_z)}, eng_{(a_x, a_y, w_z)}, max_{(a_x, a_y, w_z)}, min_{(a_x, a_y, w_z)}$ be the set of statistical features extracted from time series data. Domain specific feature set, f_D =$\{f(t_{acc}), f(t_{br}), f(t_{llc}), f(t_{rlc}), f(t_{lt}), f(t_{rt}), f(t_{non})\}$ be a set of domain-specific knowledge based features which contains the functions of aggressive and non-aggressive maneuvers.

Driving Maneuvers Classification System classifies a time series data, $t \in (S_{a_x}, S_{a_y}, S_{w_z})$ from a set of time series driving data, $T = \{t_1, t_2, \ldots t_m\}$ into a class $C_i \in C$ where m = number of test data. Therefore, the task of Driving Maneuvers Classification System is to accordingly assign t_i to C_i: <t_i, C_i>

B. FEATURE SET EXTRACTION

The performance of any deep learning algorithm heavily relies on the features applied during the training process. Since sensor fusion time series data contains meaningful patterns of driving maneuvers we consider 3 set of sensor fusion data which are S_{a_x}, S_{a_y} and S_{w_z} as our foremost features. Besides, statistical features set, f_S containing 7 statistical features of a_x, a_y and w_z i.e., total 21 statistical features. These features exhibit substantial changes in their patterns during specific maneuvers, especially, sharp slope and energy of a_x, a_y and w_z is observed during aggressive maneuvers [41]. The slope of each axis data can be defined by (10) and energy is defined by (11) [14]. Time of a particular event is another important feature. In our proposed unsupervised model, we do not consider categorical class information but in supervised model we consider categorical information i.e., class set, C as a feature. In addition, there are 7 domain-specific features for maneuver classes included in f_D.

$$Slope, S = \frac{a_{x_{slp(i)}} - a_{x_{slp(i-1)}}}{t_i - t_{(i-1)}} \tag{10}$$

$$Energy, E = \frac{{a_{x_i}}^2 + {a_{x_{(i-1)}}}^2 + \cdots + {a_{x_{i-(k+1)}}}^2}{k} \tag{11}$$

where, slope of a_x of i^{th} and $(i-1)^{th}$ point in a window is represented by $a_{x_{slp(i)}}$, a_{x_i} is a_x data of i^{th} point in the window, size of sliding window is represented by k.

We take account of these statistical features. The amount of change of sensor data during each maneuver has illustrated in Fig. 1. and based on these thresholds and data pattern change we transform the hypotheses of Section III into domain-specific binary feature vector space. Table. 2 illustrates a small fragment of domain-specific feature space.

In Table. 2, the column represents the class of maneuvers where Acc, Brake, LLC, RLC, LT, RT, and Non_agg indicate aggressive acceleration, braking, left lane change, right lane change, left turn, right turn, and non-aggressive maneuvers, respectively. If functions of an event defined by (3)– (9) is

TABLE 2. Domain-Specific Binary Feature Vector Space

TS	Acc	Brake	LLC	RLC	LT	RT	Non_agg
t_1	1	0	0	0	1	1	0
t_2	1	0	1	0	1	0	0
t_3	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...
t_n	1	0	1	0	1	0	0

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

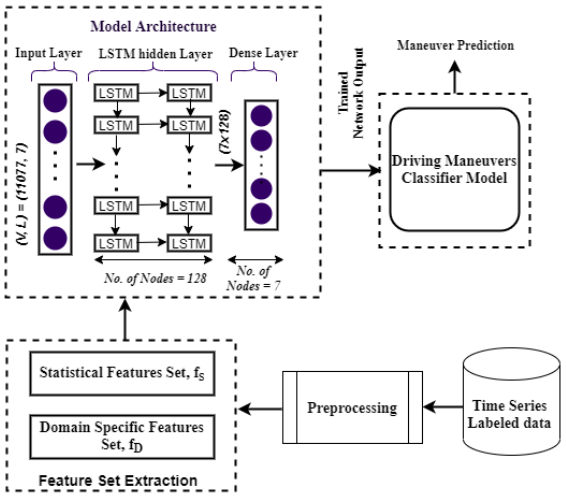

FIGURE 2. Abstract view of the proposed supervised LSTM Classifier. Driving Maneuvers Classification System automatically assigns labeled time series t_i to class C_i.

true for any time series, t_i then the column value for this event is 1, otherwise 0 where $i = 1, 2, \ldots, N$.

C. DEEP LEARNING MODEL ARCHITECTURE

Time Series classification needs to capture the long term functional dependencies between the sequences of time series and the class information by training the model with a set of known classes [46]. Among various deep learning models RNN, LSTM, GRU are suitable for the classification of sequential data. We prefer to use the LSTM model to capture the historical information between the sequence of present and past time series of driving maneuvers.

1) Supervised Model Architecture

LSTM is a variant of RNN which was proposed as a solution to the exploding and vanishing gradient problem. The model architecture comprises three major blocks: An input layer, LSTM hidden layer, and a Classification layer. An abstract view of the proposed supervised framework is depicted in Fig. 2.

- **Input Layer:** The developed LSTM model is a sequential model with multiple layers. We add the input layer to the sequential model followed by the later LSTM layers and dense layers using Keras add method. The input layer passes a few arguments which are units, activation function, input shape, and return sequences. Through the input layer, we pass 128 units or nodes which is the dimensionality of the output space and will be the input of the next layer. The input layer accepts input of shape $(t \times f)$ where t = no. of timesteps, f = no of feature, output shape is $(t \times 128)$. We consider batch size = $(\frac{1}{10})\times$ the training data size [47]. In order to handle the non-linearity, we use Rectified Linear Unit (ReLu) because it has the advantage of removing the problem of vanishing gradient faced by sigmoid and tanh activation function. Return sequences determine whether to return the last output in the output sequence or the full sequence and it has a Boolean value. We consider it as True because we want to return full sequence.
- **LSTM Hidden Layer:** We add two LSTM hidden layers to our model. Each LSTM consists of hidden units of size h. LSTM by structure has four gates which are Forget gate, Input gate, Update gate, Output gate. LSTM process an input sequence of the input vector as a pair (x_i, y_i). For each pair of (x_i, y_i) and each timestep, t a hidden vector h_t, memory vector, m_t conserved in each LSTM block. Through these vectors, the LSTM block regulates the updates and output states of blocks and eventually produce target output y_i based on past input state of input x_i. Upon receiving the output of a previous state, h_{t-1}, to keep the relevant information forget gate determines which information should be removed from previous memory vector, m_{i-1}. If we consider (13) we can see it is surrounded by a sigmoid function to express the input between 0 and 1. Input gate (i_g) determines how much new information to add from the present input to the present cell stated by (12) and through sigmoid function decides which value needs to be updated. In (14) tanh function build a new candidate vector need to added in the present cell. Update gate (c_g) use input gate, (i_g) and memory vector of previous state, $m_{i\text{-}1}$ to determine how much to write a cell which is shown by (16). Output gate (o_g) determines which information will move from the new memory vector, m_i to hidden vector h_i shown by (15). The processing is shown by (12)- (17).

$$i_g = \sigma(W_i * [h_{(t-1)}, x_t] + b_i) \tag{12}$$

$$f_g = \sigma(W_f * [h_{(t-1)}, x_t] + b_f) \tag{13}$$

$$c_g = \tanh(W_c * [h_{(t-1)}, x_t] + b_c) \tag{14}$$

$$o_g = \sigma(W_o * [h_{(t-1)}, x_t] + b_o) \tag{15}$$

$$m_t = f_g \odot m_{(t-1)} + i_g \odot c_g \tag{16}$$

$$h_t = o_g \odot \tanh(m_t) \tag{17}$$

Here σ represents the sigmoid activation function, i_g, f_g, c_g, o_g represent input gate, forget gate, update gate, output gate, respectively. W_i, W_f, W_g, W_o represent weight vector of input gate, forget gate, update gate and output gate, respectively. b_i, b_f, b_g, b_o represent bias vector of input gate, forget gate, update gate, output gate, respectively.

We used a rectified linear unit (Relu) activation function. To avoid over-fitting a dropout layer is being introduced after each hidden layer with a dropout ratio of 20% [48]. Dropout is a phenomenon of a machine learning model that performs better on the training data compared to the test data. At every iteration, 80% of neurons are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

randomly selected by the model to pass their output from the first hidden layer to the second hidden layer which further generates a new vector [48].

- **Dense Classification Layer:** We add a dense layer at the end of the model to make our model more robust. The number of neurons in the dense layer is equal to the number of classes. For our model, the number of neurons in the dense layer is seven since we want to predict seven classes of maneuvers.
  Eq. (18) corresponds to the cross-entropy loss function [48] that we used to train the model. We use categorical cross-entropy as we are dealing with a multi-class categorical classification problem. Here i subscript indicates i^{th} time series input, T is size of output time series and t_i represents actual driving class of i^{th} time series.

$$Loss(y, y_{pred}) = -\frac{1}{T}\sum_{i=1}^{T}(t_i \cdot log(y_{pred})) \qquad (18)$$

This module determines the class of driving maneuvers from new time series data. For classification, new unlabeled data is feed to the developed LSTM classifier. From the feature vector of the unlabeled test data the trained classifier predicts the probability of unlabeled data to belong to a particular class. In Section IV-A2 we discuss the training of an autoencoder model to learn the compressed feature vector of unlabeled data and transfer the learning to Driving Maneuvers Classifier Model to predict the class of the unlabeled data.

2) Unsupervised Model Architecture

An LSTM Autoencoder is an application of an autoencoder for sequence data that uses an Encoder-Decoder LSTM architecture. A very well-known unsupervised implementation of LSTM autoencoder is that once fit in the model, the encoder part can be used to encode and compressed the sequence data. In turn, the encoded or compressed data can be used in data visualization or as a feature vector input to another supervised learning model. Considering its training mechanism which is based on supervised learning, it is also referred to as a self-supervised learning model. Since the entire process comprises of unsupervised and supervised learning model, it is sometimes called semi-supervised learning. Alike other typical autoencoders, our developed encoder is a part of a broader LSTM autoencoder model that endeavors to regenerate the input.

The proposed unsupervised LSTM autoencoder model architecture comprises three major blocks: Encoder, Code or Latent Space Representation, and Decoder. The structure of the proposed stacked autoencoder is depicted in Fig. 3.

- **Encoder Layer:** In the encoding phase, the model learns a compressed latent representation of the input by mapping the input to the hidden layer. Given an unlabeled time series input x_t, where $t = 1, 2, \ldots N$. The encoder function takes input x_t and provides encoded vector, h_1 by (19) and shown in Fig. 3. h_1 becomes the input of the next layer to produce encoded vector h_2 shown in (20). The number of cells in the input layer of the LSTM autoencoder is equal to the timesteps, t. Through the input layer of the LSTM autoencoder, we pass few arguments such as the number of units or dimension of output space in the subsequent layer, activation function, input shape, return sequences. Return sequences as true will make each cell per timestep emit a signal. Signal emitted from a timestep cell of the previous layer, l_1 is transferred to the cell of the same timestep in the following layer. The flow diagram of the proposed LSTM autoencoder is illustrated in Fig. 4. The first encoding layer takes input data of shape $(t \times f)$ where t = number of timesteps and f = number of features and t number of timesteps each will outputs u number of units or output dimension space as we consider return sequence as True. The second encoding layer, l_2 receives $t \times u$ input from l_1 and reduces the feature size v as our goal is to compressed the input dimension. In l_2 we consider return sequence is also True, so the output shape is $t \times v$. We use activation function Exponential Linear Unit (ELU). ELU has a similar function as ReLU except negative input. It tends to converge the cost to zero faster than other activation functions. It becomes smooth slowly while ReLU becomes smooth sharply. ELU provides better result for our model.

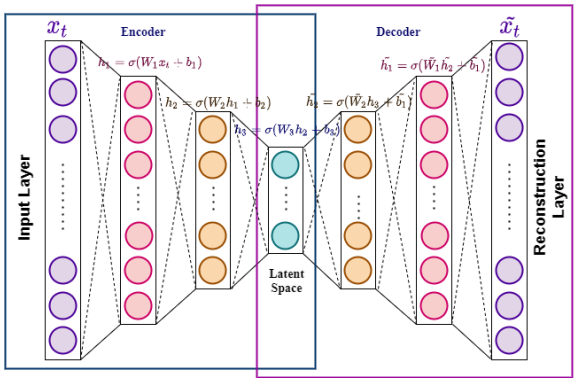

FIGURE 3. Architecture of the proposed Autoencoder

$$h_1 = \sigma(W_1 x_t + b_1) \qquad (19)$$

$$h_2 = \sigma(W_2 h_1 + b_2) \qquad (20)$$

Here, W_1, b_1 is the weight vector and bias vector of l_1 and W_2, b_2 represents the weight vector and bias vector of l_2, respectively.

- **Latent Space Representation:** Layer l_3 outputs a reduced dimension of size z and return sequence is False. Hence, this layer will output an encoded feature vector of size $1 \times z$ of the input data. We extract the encoded feature vector, h_3 and transfer the learning in our supervised LSTM network. We add a RepeatVector method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

which creates an interconnection between encoder and decoder. The argument of RepeatVector method is t and it imitates the feature vector t times. In (21), h_3 is the encoded vector or latent space, W_3 and b_3 is the weight vector and bias vector of code layer l_3, respectively.

$$h_3 = \sigma(W_3 h_2 + b_3) \tag{21}$$

- **Decoder:** In the decoding phase, the model recreates the target from the compressed latent representation during the encoding phase. Decoder layers unfold the encoding layers and stacked in reverse order of encoding layers. The layers of decoder l_5, l_6, l_7 are mirror image of encoder layer l_3, l_2 and l_1, respectively. $f(\tilde{x}_t)$ is the reconstruction function of x_t.

$$\tilde{h}_2 = \sigma(\tilde{W}_2 h_3 + \tilde{b}_2) \tag{22}$$

$$f(\tilde{x}_t) = \tilde{h}_1 = \sigma(\tilde{W}_1 \tilde{h}_2 + \tilde{b}_1) \tag{23}$$

In (22) $\tilde{h}_2, \tilde{W}_2, \tilde{b}_2$ represents the decoded vector, weight vector and bias vector of l_5 layer, respectively. Similarly, in (23) $\tilde{h}_1, \tilde{W}_1, \tilde{b}_1$ represents the decoded vector, weight vector and bias vector of l_6 layer, respectively.
We added a TimeDistributed wrapper with dense layer which takes an argument equal to the number of features, f. This layer, l_8 outputs a vector of length equal to the number of features outputted from the l_7 layer. Layer l_7 outputs u features alike encoding layer l_1. Hence, TimeDistributed layer, l_8 takes a u long vector and duplicates it f times. A dropout layer is being introduced after each hidden layer with a dropout ratio of 30% [48]. At every iteration, 70% of neurons are randomly chosen by the autoencoder model to pass their output from the one hidden layer to the next hidden layer.
Eq. 24 presents the mean squared error loss function [48] that we use to train the model with the encoded feature vector. The loss is the mean of the squared differences between the target variable and the predicted value.

$$Loss(y, y_{pred}) = -\frac{1}{N}\sum_{i=1}^{N}(y - y_{pred}) \tag{24}$$

In (24), i subscript indicates i^{th} input, N is number of input time series, y, y_{pred} represents target variables and predicted values, respectively.

3) Semi-Supervised Driving Maneuvers Classifier Model

The goal of the semi-supervised driving maneuvers classifier model module is to determine the class of driving maneuvers from sensor fusion data that has never seen by the model. Initially, new test data is fed to the unsupervised LSTM autoencoder model for latent representation learning where it passes through the encoding and decoding process. In order to train the supervised LSTM classifier with unsupervised learning, this encoded feature vector is transferred to the supervised LSTM classifier as input and in turn, the classifier can predict the classes of the test data.

V. RESULT AND ANALYSIS

In this section, we describe the used dataset, experimental environment setup, hyperparameters settings for both supervised and semi-supervised model. Finally, we explain the experimental results for both models.

A. DATASET DESCRIPTION

The dataset [49] used in this experiment contains 156512 time-series data from 4 trips conducted by two drivers who have an expertise of 15 years in driving and the roads were smooth. Time series was recorded by a smartphone application where accelerometer and gyroscope sensors were preinstalled and the smartphone was placed on the windshield in a continuous stable position. During the event, the front view of the subject vehicle was also recorded by a camera which further helped to do manual labeling of the driving maneuvers. The drivers were told to do a specific maneuver and no pre-training was provided.

B. EXPERIMENTAL ENVIRONMENT SET UP

The aim of this experiment is to identify a suitable hyperparameter combination to optimize the developed deep learning model and analysis the efficiency of the model over other related developed classifiers. We use jupyter notebook to conduct the experiments. The architectures are being implemented using Keras == 2.3.0 framework with Tensorflow == 2.0 (CPU) backend in Python == 3.8.3. For statistical analysis and visualization Matplotlib 3.3.3 and Seaborn 0.11.1 are used. Numpy 1.19.4 and Scikit-learn 0.24.0 are used for scientific computation. The model is trained on CPU instances with RAM 8 GB and a core i-5 processor.

C. HYPERPARAMETER OPTIMIZATION

Hyperparameter has a significant impact on the performance of the model. Network architecture (the number of neurons, the number of layers) and the process of training (batch size, learning rate, optimizer) is being determined by hyperparameter. In our work, we used two different settings for the supervised and unsupervised model.

1) Hyperparameter of Supervised Model

Two LSTM hidden layers have been used with 128 units for each. The hyperparameter setting such as batch size, dropout rate, optimizer, learning rate, and the number of epochs of the proposed supervised model is listed in Table. 3. To discover the optimal hyperparameter we iterate through the following hyperparameter space. The proposed supervised LSTM model is being trained with an optimized combination of hyperparameters.

2) Hyperparameter of Unsupervised Model

The output dimension of the input layer is 31 and in two hidden layers 20 and 10 hidden units, respectively are used. The hyperparameter setting for the unsupervised model such as batch size, dropout rate, epochs, learning rate has been

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

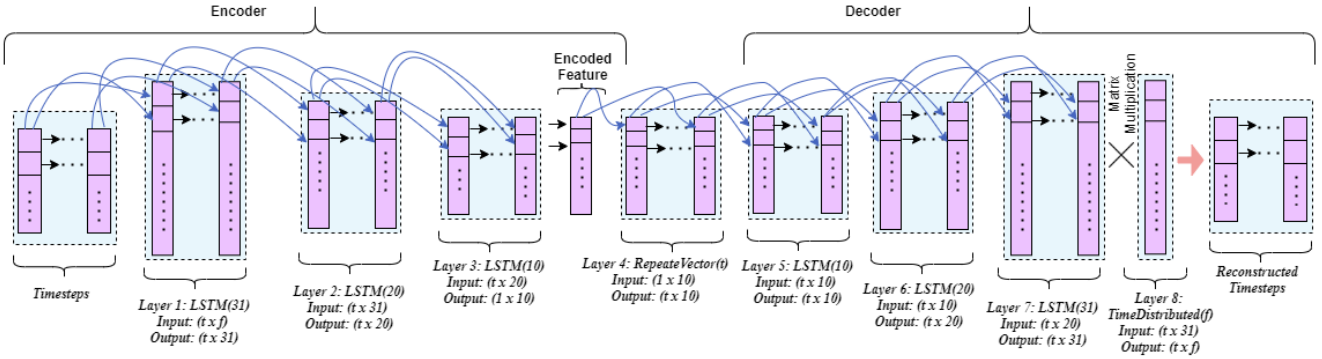

FIGURE 4. Flow diagram of proposed Encoder-Decoder LSTM architecture

TABLE 3. Hyperparameter settings for Supervised Model

Hyperparameters	Hyperparameter space	Optimal value
Batch Size	10, 20, 50, 100, 500, 800, 1000, 12521	886
Dropout	0.1, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75	0.2
Optimizer	SGD, RMSprop, Adam, Adadelta	Adam
Learning Rate	0.9, 0.6, 0.1, 0.09, 0.06, 0.01, 0.009, 0.006, 0.001, 0.0009, 0.0006, 0.0001, 0.00001	0.001
Number of Epochs (for stand alone Supervised Model)	10, 50, 100, 150, 300, 500, 1000, 1800, 2000, 3000	500
Number of Epochs (for Semi-Supervised Model)	10, 50, 100, 150, 300, 500, 1000, 15000	1000

TABLE 4. Hyperparameters Setting for Unsupervised Model

Hyperparameters	Hyperparameter space	Optimal value
Batch Size	10, 50, 100, 200, 500, 874, 1000	14086
Dropout	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75	0.3
Optimizer	SGD, RMSprop, Adam, Nadam	Adam
Learning Rate	0.9, 0.6, 0.3, 0.1, 0.09, 0.06, 0.03, 0.01, 0.009, 0.006, 0.003, 0.001, 0.0009, 0.0006, 0.0003, 0.0001, 0.00001, 0.000001	0.001
Number of Epochs	10, 50, 100,120, 150, 300, 500	300

specified in Table 4. To find the optimized hyperparameter combination we iteratively train the model through the hyperparameter setting. The proposed unsupervised autoencoder model is being trained with the optimal hyperparameters and provides lower loss and the best performance after transfer the encoded vector to the supervised model.

D. RESULTS AND DISCUSSIONS

The results of the proposed work is discussed in two fold. At first, we discuss the results of the proposed supervised model. Then we analyse the result of semi-supervised model. Besides, we compare the proposed work with other related works. A 10-fold Cross-Validation is applied to evaluate the skill of both stand alone Supervised, Unsupervised and Semi-Supervised model. For each fold the models are trained with 9 fold of the dataset and are validated on remaining fold of the dataset.

1) Results of Supervised Model

The performance of the standalone Supervised model has been evaluated by accuracy, loss, precision, recall, F1-score and ROC curve on test dataset and performance for all the folds are listed in Table 5. The best supervised model is achieved in the 10^{th} fold shown in bold. The best performance of the model on test data is computed accuracy, loss, precision, recall and F1-score is 0.9810, 0.1609, 0.9767, 0.9822 and 0.9794, respectively.

A 7×7 confusion matrix of the best standalone Supervised model is shown in Table. 6 where the rows and columns represent actual class and predicted class, respectively when 7 is the number of target classes. This matrix compares the actual target values with that of predicted by the supervised model.

The comparison of evaluation scores of the proposed best Supervised model trained with and without f_D are listed in Table 7.

The variation of accuracy and loss over the number of epochs for the training and validation dataset of the best supervised model is demonstrated by Fig. 5 and Fig. 6, respectively. Initially, the training and validation accuracy increases simultaneously, but after 100 epochs increasing rate becomes slower. On the other hand, training and validation loss decreases to 100 epochs, after that the increment rate becomes slower. At 500 epochs, the model provides the best training and validation accuracy which is 99.98% and 98.10%, and training and validation loss is 06.84% and 16.09%. Fig. 7

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

TABLE 5. Evaluation Metrics of 10 fold Cross-Validation of standalone Supervised Model trained with f_S and f_D

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
Train Accuracy	0.9990	0.9613	0.6693	0.9075	0.7154	0.9570	0.9988	0.9963	0.9724	**0.9998**
Train Loss	0.0039	0.1049	0.8818	0.2541	0.7886	0.1197	0.0042	0.0106	0.0788	**0.0684**
Test Accuracy	0.9666	0.9404	0.6832	0.9025	0.6877	0.9404	0.9602	0.9557	0.9250	**0.9810**
Test Loss	0.1619	0.2117	0.8849	0.2920	0.8302	0.1717	0.1871	0.2114	0.2280	**0.1609**
Precision	0.9599	0.9309	0.7437	0.9129	0.6723	0.9327	0.9584	0.9500	0.9161	**0.9768**
Recall	0.9591	0.9261	0.5796	0.8387	0.5850	0.9405	0.9503	0.9490	0.9212	**0.9822**
F1-Score	0.9594	0.9267	0.5988	0.8653	0.5969	0.9354	0.9541	0.9491	0.9184	**0.9794**

TABLE 6. Confusion Matrix of Proposed best Supervised Model trained with f_S and f_D

Actual Class	Predicted Class						
	ACC	Brake	LLC	LT	RLC	RT	Non _Agg
ACC	214	3	0	0	0	0	3
Brake	0	145	0	1	0	0	0
LLC	0	0	43	0	0	0	0
LT	1	0	0	192	0	0	2
RLC	1	0	0	0	54	0	1
RT	1	1	0	0	0	192	1
Non_Agg	2	0	2	0	2	0	247

TABLE 7. Evaluation score of proposed best Supervised Model trained with and without f_D

Evaluation Metric	Score	
	Proposed Model trained with f_D	Proposed Model trained without f_D
Train Accuracy	0.9998	0.9980
Test Accuracy	0.9810	0.9620
Train Loss	0.0684	0.4824
Test Loss	0.1609	0.3047
Precision	0.9768	0.9645
Recall	0.9822	0.9588
F1-score	0.9794	0.9615

depicted the Receiver Operating Characteristic (ROC) curve of Supervised Driving Maneuvers Classification.

To analyze the performance of the model over other classical or machine learning classifier models we compare its performance with other related works which is listed in Table 8. From Table. 8 it is clearly observed that deep learning methods perform better than other methods. Among deep learning models, LSTM and GRU perform better than the SimpleRNN. In comparison to Paper [21], our proposed supervised model outperforms. We tested our model by training with only statistical features and also along with domain-specific features. The model provides the best performance while training with statistical features along with domain-specific features. It takes 500 epochs and two hours to train with statistical features along with domain-specific features for best performance while 2000 epochs and eight hours for training without domain-specific features. For training with an unsupervised encoded feature vector, the proposed supervised model needs to train 1000 epochs to get the best performance.

TABLE 8. Comparison of the proposed work with other Supervised techniques

Paper	Classifier/Model	Accuracy
Proposed Supervised Method	LSTM	98.10%
Proposed Semi-Supervised Method	AE-LSTM	88.35%
Alvarez-Coello *et al.* [22]	RNN	78.59%
Carvalho *et al.* [21]	LSTM and GRU	>95%
	SimpleRNN	70%

As we train the best supervised model with best unsupervised latent space representation of driving data the performance of the Semi-Supervised model equally depends on the performance of unsupervised latent space learning of unlabeled data and the supervised classifier model. The results of unsupervised cross-validation is listed in Table 9. Autoencoder model provides by nature a lossy reconstruction of data. When the proposed LSTM autoencoder learns the unsupervised latent representation there is a small amount of loss involves discussed in Section V-B2 which further degrades the performance of the supervised model. The proposed semi-supervised model provides accuracy, precision, recall, F1 score of 88.35%, 85.99%, 86.37%, 86.15%, respectively on test data which is less than that of the proposed supervised model. However, as the Semi-Supervised approach can be used for the classification of unlabeled data it is more useful than supervised techniques. We also proposed a few future approach to improve the performance of the proposed semi-supervised approach.

2) Results of Semi-Supervised Model

We apply 10-fold Cross-Validation to build the best unsupervised encoded latent space feature vector for various LSTM autoencoders architectures. The architectures and corresponding evaluation scores are listed in Table 13. The Mean-Squared-Error (MSE) and Root-Mean-Squared-Error (RMSE) loss with the best LSTM-AE architecture model for each fold is presented in Table 9 and the best developed unsupervised model is found in 6^{th} fold. This encoded feature vector found from the 6^{th} fold is transferred to the supervised model as input. This supervised model is trained applying stratified 10-fold cross-validation. The evaluation score of Semi-Supervised model are presented in Table 10 and it is clear from Table 10 fold 1 provides the best semi-supervised Model.

Fig. 8 illustrates the train and validation loss during the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

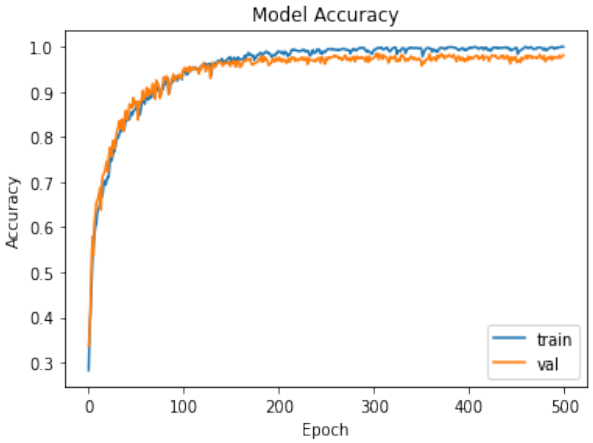

FIGURE 5. Accuracy vs number of epochs applying f_S and f_D

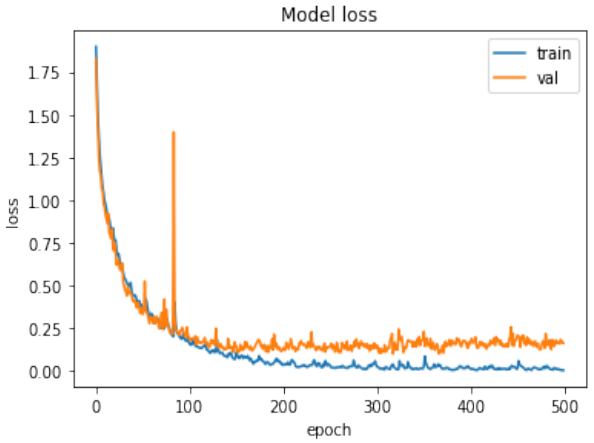

FIGURE 6. Loss vs number of epochs applying f_S and f_D

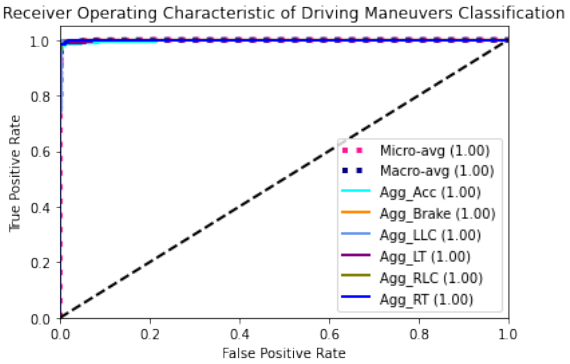

FIGURE 7. Receiver Operating Characteristic curve of Supervised Driving Maneuvers Classification.

training process of the LSTM Autoencoder model over the number of epochs for the training and validation dataset. Initially, the training and validation loss decreases simultaneously, but after 50 epochs decreasing rate becomes slower. At 300 epochs, the AE model provides the MSE and RMSE loss is 82.51% and 9.08%, respectively. The encoded feature vector of the autoencoder is transferred to and trained the supervised LSTM model and after that train and validation accuracy and loss during the training process of the LSTM model over the number of epochs for the training and validation dataset is depicted by Fig. 9 and 10, respectively. In Fig. 9, the training and validation accuracy increases smoothly until 600 epochs and then increases slowly until 1000 epochs. On the other hand, in Fig. 10, the training and validation loss decreases smoothly until 600 epochs and then decreases slowly until 1000 epochs. Fig. 11 depicts the Receiver Operating Characteristic (ROC) curve of Semi-Supervised Driving Maneuvers Classification. The ROC curve depicted that for LT the curve area is 1 and for Acceleration, brake, LLC, RLC, RT and Non Aggressive the curve area is 0.99, 0.98, 0.97, 0.97, 0.97 and 0.98, respectively. A 7×7 confusion matrix of the best Semi-Supervised model is shown in Table 11.

Another evaluation metric of time series multiclass classification Mean Per Class Error (MPCE) has been proposed by Wang *et al.* in [47]. Mean Per Class Error is the calculated average of error occurred in the prediction of each class by the classifier model. The Per Class Error can be defined by (25).

$$Per - Class - Error(PCE) = \frac{1 - accuracy}{NumberofClasses} \tag{25}$$

The Per Class Error (PCE) of the best supervised model after training with and without domain specific features and Semi-Supervised model are listed in Table 12. The Mean Per Class Error (PCE) of the proposed standalone Supervised model is 0.005415 and Semi-Supervised model is 0.033135 applying domain-specific features with statistical features. For all the class accuracy are above 99% and for non-aggressive class accuracy is above 98%. On the other hand, the Mean Per Class Error (MPCE) of the proposed Supervised model is 0.049465 without applying domain-specific features with statistical features. Besides, all of the class accuracy are significantly lower than the class accuracy of our proposed supervised and semi-supervised method trained with domain features. Therefore, it indicates that using domain-specific knowledge helps to train the model and improves the performance.

The architecture of various LSTM autoencoders and corresponding metrics is presented in Table 13. We find that increasing the number of layers in the autoencoder structure increases the training time. Also, a higher number of units in the layers increases the training time. On the contrary, a too small number of units restrict the autoencoder to learn all the significant hidden features. Hence, we investigate a structure of stacked autoencoder that has a minimum number of layers and a minimum number of units in each layer to learn the hidden representation. We add 10 units in the encoded layer so that it will be the size of the feature vector of the supervised model and need less time during the supervised

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

TABLE 9. MSE and RMSE of 10 fold Cross-Validation of Unsupervised Model

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
MSE	111.65	101.15	150.73	110.33	137.94	**82.51**	144.40	157.22	122.46	246.79
RMSE	10.56	10.06	12.27	10.50	11.74	**9.08**	12.01	12.53	11.06	15.70

TABLE 10. Evaluation Metrics of 10-fold Cross-Validation of Semi-Supervised Model

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
Train Accuracy	**0.9134**	0.8867	0.9190	0.8797	0.8859	0.9171	0.8898	0.9004	0.9185	0.8844
Train Loss	**0.2262**	0.2931	0.2122	0.3099	0.2967	0.2130	0.2796	0.2508	0.2089	0.2946
Test Accuracy	**0.8835**	0.8547	0.8637	0.8330	0.8213	0.8628	0.8529	0.8609	0.8735	0.8536
Test Loss	**0.4779**	0.5682	0.5768	0.4652	0.5407	0.5894	0.6373	0.4771	0.4339	0.4447
Precision	**0.8599**	0.8630	0.8358	0.8125	0.8029	0.8420	0.8324	0.8382	0.8625	0.8281
Recall	**0.8638**	0.8179	0.8181	0.7947	0.7857	0.8353	0.8343	0.8321	0.8465	0.8333
F1-Score	**0.8616**	0.8359	0.8253	0.8017	0.7898	0.8383	0.8327	0.8344	0.8531	0.8291

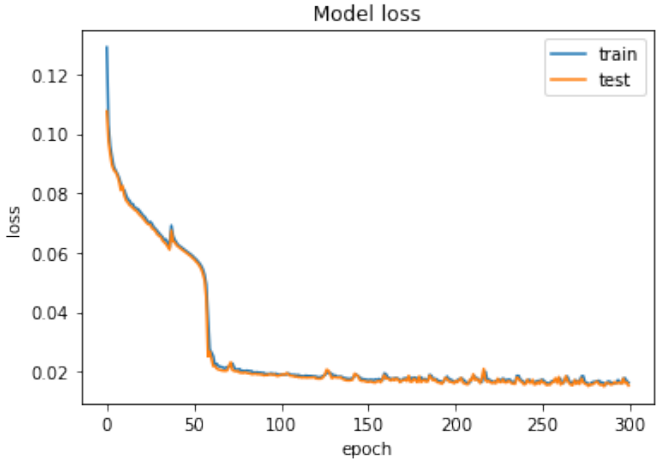

FIGURE 8. Loss variation of LSTM Autoencoder during the training/validation process

TABLE 11. Confusion Matrix of Proposed Semi-supervised Model trained with encoded feature vectors

Actual Class	Predicted Class						
	ACC	Brake	LLC	LT	RLC	RT	Non _Agg
ACC	199	13	0	1	0	4	3
Brake	12	124	0	2	1	6	1
LLC	3	1	33	1	0	0	5
LT	0	1	3	181	3	4	3
RLC	1	1	0	2	47	4	1
RT	3	0	4	10	3	167	8
Non_Agg	5	5	7	3	0	5	228

training process. Besides, increasing the dropout rate of the autoencoder model increases training accuracy and testing loss of a supervised model. Hence, we chose a dropout rate of 30% for optimal training and testing performance.

Though there are a few work done by adopting semi-supervised approach, if we do a comparative analysis of [37] and [42] with the performance of our proposed Semi-Supervised method none of them have provided accuracy and error of each class. Hence, we can not do any direct comparison. Moreover, [37] also considered only two classes which are normal and aggressive. They found 86% accuracy using semi-supervised SVM which is a little lower than that of our semi-supervised model. However, classifying multi-class driving maneuvers increases the number of classifiers and computational complexity as well [42]. In [42], accuracy has been computed for motion, velocity, turning were 87.3%, 78.4% and 76.9%, respectively. In motion category, sub classes were move and stop which were very easy to find out. In velocity category, sub classes were acceleration, deceleration, constant speed and in turning category, sub classes were left/right turning, left/right curving which was most difficult to classify. By our proposed semi-supervised approach all of the classes provide accuracy greater that 95%.

E. DISCUSSION

We implement an unsupervised LSTM Autoencoder to learn encoded vector and a supervised LSTM classifier model to classify the driving maneuvers of the labeled encoded feature vector. We also evaluate the performance of our proposed standalone supervised LSTM network for classifying maneuvers class from labeled data. Our semi-supervised model provides lower performance than supervised model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089660, IEEE Access

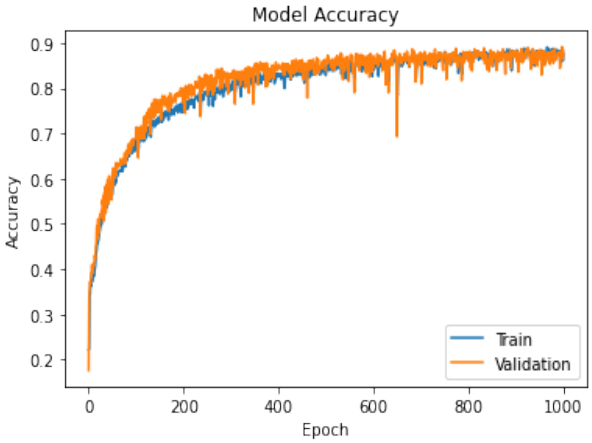

FIGURE 9. Accuracy variation of Supervised LSTM Model during the training/validation with encoded vector

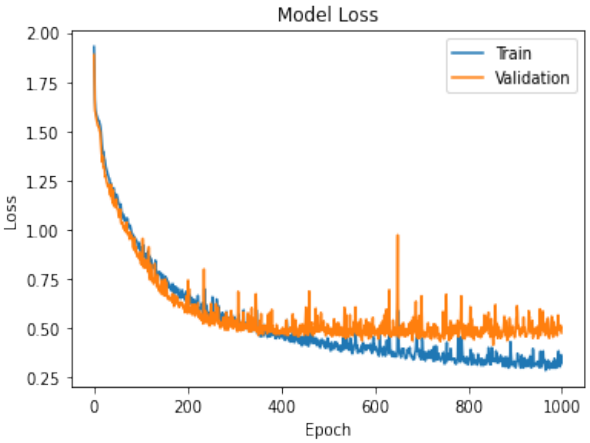

FIGURE 10. Loss variation of Supervised LSTM Model during the training/validation with encoded vector

TABLE 12. Per Class Accuracy and Error of Proposed Supervised Model trained with and without f_D and Semi-Supervised Model

Class	Supervised Model trained with f_D		Supervised Model trained without f_D		Semi-Supervised Model trained with f_D	
	Per Class Accuracy	Per Class Error	Per Class Accuracy	Per Class Error	Per Class Accuracy	Per Class Error
Aggressive Acceleration	0.9901	0.0099	0.9340	0.0659	0.9594	0.0406
Aggressive Braking	0.9955	0.0045	0.9673	0.0326	0.9612	0.0388
Aggressive LLC	0.9982	0.0018	0.8955	0.1044	0.9784	0.0216
Aggressive RLC	0.9963	0.0036	0.9292	0.0707	0.9856	0.0144
Aggressive LT	0.9963	0.0036	0.9811	0.0188	0.9702	0.0297
Aggressive RT	0.9973	0.0027	0.9926	0.0073	0.9540	0.0460
Non Aggressive	0.9883	0.0117	0.9538	0.0461	0.9594	0.0406

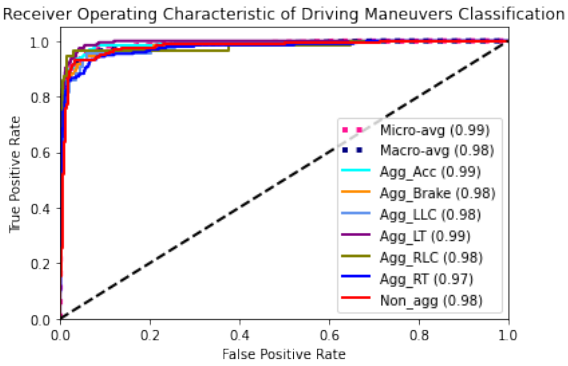

FIGURE 11. Receiver Operating Characteristic curve of Semi-Supervised Driving Maneuvers Classification

TABLE 13. Architecture of Autoencoder and corresponding metrics of Semi-Supervised Model

AE Structure	Unsupervised Model	Supervised Model			
	RMSE	Train Accuracy	Test Accuracy	Train Loss	Test Loss
31-20-10-20-31	**9.08**	**0.9134**	**0.8835**	**0.2262**	**0.4779**
40-16-40	12.46	0.8357	0.8043	0.4525	0.6438
31-24-16-8-16-24-31	12.42	0.8522	0.8163	0.3902	0.5509
55-16-55	12.64	0.8118	0.8023	0.5161	0.5929
50-16-50	13.51	0.7920	0.7725	0.5617	0.6692
64-18-64	13.99	0.7546	0.7473	0.6757	0.7443
64-16-64	15.68	0.7179	0.7089	0.5452	0.7001
64-31-64	15.70	0.6758	0.6677	0.8535	0.9021
31-16-31	16.31	0.6771	0.6651	0.8473	0.8871

A high number of training data causes data redundancy [42]. Besides, autoencoder provides a lossy output by nature; therefore the encoded vector is not the same as the feature vector of Supervised model. Consequently, the performance of semi-supervised model degrades than supervised model. We also observe by plotting few ground truth of data was mislabeled by data analysts whose pattern does not match with data change rule according to their labels. We did not manually update the ground truth label of the data as done in [42] to reduce error occurred by algorithm. The mislabeling does not affect supervised model's accuracy because supervised model is being trained with label information and its performance is tested only on labeled data. On the other hand, unsupervised model learns from all the data disregarding the label information. This learning is free from human bias. Unsupervised encoded learning of labeled data trains the supervised model. This training is different from the training of standalone supervised model. After training performance

of supervised model is tested on test data labeled by human.

VI. CONCLUSION

In this work, we present a Semi-supervised transfer learning approach for the classification of driving maneuvers from sensor fusion data. To learn the unlabeled time series data, we develop an LSTM autoencoder and transfer the encoded feature vector to train the supervised LSTM model. We train our model with both statistical features as well as domain-specific features and found that training the model in this way improves the precision and performance of the model and reduces time complexity. The proposed supervised model outperforms the other existing machine learning and deep learning classifier. The results of the experiments exhibit a semi-supervised technique for the classification of unlabeled time-series driving data. In this work, we have separate losses in both the unsupervised and supervised models. In the future, we will try to combine and so that reduce the loss of both neural network models. Besides, we have a plan to add a neural attention mechanism with the proposed model to focus on a subset of features of the time series dataset.

REFERENCES

[1] N. Kattukkaran, A. George, and T. M. Haridas, "Intelligent accident detection and alert system for emergency medical assistance," in *Proc. ICCCI,* Coimbatore, India, 2017, pp. 1–6. Available: https://doi.org/10.1109/ICCCI.2017.8117791

[2] R. K. Kodali, and S. Sahu, "MQTT based vehicle accident detection and alert system," in *Proc. iCATccT,* Tumkur, India, 2017, pp. 186–189. Available: https://doi.org/10.1109/ICATCCT.2017.8389130

[3] P. Nath, and A. Malepati, "IMU based Accident Detection and Intimation System," in *Proc. IEMENTech,* Kolkata, India, 2018, pp. 1–4. Available: https://doi.org/10.1109/IEMENTECH.2018.8465309

[4] S. Sarker, M. S. Rahman, and M. N. Sakib, "An Approach Towards Intelligent Accident Detection, Location Tracking and Notification System," in *Proc. ICTP,* Dhaka, Bangladesh, 2019, pp. 1–4. Available: https://doi.org/10.1109/ICTP48844.2019.9041759

[5] E. Ohn-Bar, A. Tawari, S. Martin, and M. M. Trivedi, "Predicting driver maneuvers by learning holistic features," in *Proc. IEEE Intelligent Vehicles Symposium,* Dearborn, MI, USA, June, 2014, pp. 719-724.

[6] B. Morris, A. Doshi, and M. Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In IEEE International Vehicle Symposium Proceedings, 2011.

[7] A. Jain, H. S. Koppula, S. Soh, B. Raghavan, A. Singh, and A. Saxena, "Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture," 2016. Available: arXiv preprint arXiv:1601.00740.

[8] M. G. Ortiz, J. Schmüdderich, F. Kummert, and A. Gepperth, "Situation-specific learning for ego-vehicle behavior prediction systems," in *Proc. ITSC* Washington, DC, USA, October, 2011, pp. 1237-1242.

[9] P. Angkititrakul, R. Terashima, and T. Wakita, "On the use of stochastic driver behavior model in lane departure warning," in *IEEE Transactions on intelligent transportation systems,* vol. 12, no. 1, 2010, pp. 174-183.

[10] G. Xu, L. Liu, Y. Ou, and Z. Song, "Dynamic modeling of driver control strategy of lane-change behavior and trajectory planning for collision prediction," in *IEEE Transactions on Intelligent Transportation Systems,* vol. 13, no. 3, 2012 pp. 1138-1155.

[11] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model," in *Proc. 2012 IEEE Intelligent Vehicles Symposium,* Alcala de Henares, Spain, June, 2012, pp. 1162-1167.

[12] CAN Bus. Available: https://en.wikipedia.org/wiki/CAN_bus

[13] Micro Electro Mechanical System. Available: https://en.wikipedia.org/wiki/Microelectromechanical_systems

[14] M. Wu, S. Zhang, and Y. Dong, "A novel model-based driving behavior recognition system using motion sensors," in *Sensors,* vol. 16, no. 10, 2016, pp. 1746.

[15] A. Sathyanarayana, S.O. Sadjadi, J.H.L. Hansen, "Leveraging Sensor Information from Portable Devices towards Automatic Driving Maneuver Recognition," in *Proc. ITSC,* Anchorage, AK, USA, September, 2012, pp. 660–665.

[16] C. Saiprasert, S. Thajchayapong, T. Pholprasit, and C. Tanprasert, 2014, November, "Driver behaviour profiling using smartphone sensory data in a V2I environment," in *Proc. ICCVE,* Vienna, Austria, 2014, pp. 552-557. Available: https://doi.org/10.1109/ICCVE.2014.7297609

[17] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena, "Recurrent neural networks for driver activity anticipation via sensory-fusion architecture," in *Proc. ICRA,* Stockholm, Sweden, May, 2016, pp. 3118-3125.

[18] D. A. Johnson, and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *Proc. ITSC,* Washington, DC, USA, 2011, pp. 1609-1615. Available: https://doi.org/10.1109/ITSC.2011.6083078

[19] G. Castignani, R. Frank, and T. Engel, "Driver behavior profiling using smartphones," in *Proc. ITSC,* The Hague, Netherlands, 2013, pp. 552-557. Available: https://doi.org/10.1109/ITSC.2013.6728289

[20] 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," in *PLoS one,* vol. 12, no. 4, 2017, pp. e0174959.

[21] E. Carvalho, B. V. Ferreira, J. Ferreira, C. De Souza, H. V. Carvalho, Y. Suhara, G. Pessin,*et al.*, "Exploiting the use of recurrent neural networks for driver behavior profiling," in *Proc. IJCNN,* Anchorage, AK, USA, 2017, pp. 3016-3021.

[22] D. Alvarez-Coello, B. Klotz, D. Wilms, S. Fejji, J. M. Gómez, and R. Troncy, "Modeling dangerous driving events based on in-vehicle data using Random Forest and Recurrent Neural Network," in *Proc. IV,* Paris, France, France, June, 2019, pp. 165-170.

[23] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, and P. Loos, "Time series classification using deep learning for process planning: A case from the process industry,"*Procedia Computer Science,* vol. 114, pp. 242-249, 2017.

[24] D. E. Rumelhart, R. J. Williams, G. E. Hinton, "Learning internal representations by error propagation," *California Univ San Diego La Jolla Inst for Cognitive Science,* 1985.

[25] D. Charte, F. Charte, M. J. del Jesus, and F. Herrera, "An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges,"*Neurocomputing,* 2020.

[26] R. Bellman, and R. Kalaba, "On adaptive control processes,"*IRE Transactions on Automatic Control,* vol. 4, no. 2, pp. 1-9, 1959.

[27] L. A. Zadeh, G. J. Klir, and B. Yuan, "Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers," in *World Scientific,* Vol. 6, 1996.

[28] C. Schwarz, "Time Series Categorization of Driving Maneuvers Using Acceleration Signals," in *Driving Assessment Conference,*, Iowa Research Online, USA, 2017.

[29] P. Patel, E. Keogh, J. Lin, and S. Lonardi, "Mining motifs in massive time series databases," in *Proc. IEEE International Conference on Data Mining,* Maebashi City, Japan, 2002, pp. 370-377.

[30] C. C. M. Yeh, Y. Zhu, L. Ulanova, and N. Begum, Y. Ding, H. A. Dua, *et. al*"Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *Proc. ICDM,* Barcelona, Spain, 2016, pp. 1317-1322.

[31] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," in *Proc. IV,* Gold Coast, QLD, Australia, 2013, pp. 1040-1045. Available: https://doi.org/10.1109/IVS.2013.6629603

[32] C. Cortes, and V. Vapnik, "Support-vector networks," in *Machine learning,* vol. 20, no. 3, September, 1995, pp. 273-297.

[33] J. MacQueen, "Some methods for classification and analysis of multivariate observations,"in *Proc. Berkeley symposium on mathematical statistics and probability,*, vol. 1, no. 14, University of California Press, 1967, pp. 281-297.

[34] J. Cervantes-Villanueva, D. Carrillo-Zapata, F. Terroso-Saenz, M. Valdes-Vela, and A. F. Skarmeta, "Vehicle maneuver detection with accelerometer-based classification," in *Sensors,* vol. 16, no. 10, 2016, pp. 1618.

[35] H. T. Kam, "Random decision forest,"in *Proc. International Conference on Document Analysis and Recognition,* vol. 1416, Montreal, Canada, August, 1995, pp. 278282.

[36] J. Pearl, "Bayesian netwcrks: A model cf self-activated memory for evidential reasoning," in *Proc. Conference of the Cognitive Science Society,* University of California, Irvine, CA, USA, August, 1985, pp. 15-17.

[37] W. Wang, J. Xi, A. Chong, and L. Li, "Driving style classification using a semisupervised support vector machine," in *IEEE Transactions on Human-Machine Systems,* vol. 47, no. 5, 2017, pp. 650-660.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *nature,* vol. 323, no. 6088, 1986, pp. 533-536.

[39] S. Hochreiter, and J. Schmidhuber, "Long short-term memory,"in *Neural computation,* vol. 9, no. 8, 1997 pp. 1735-1780.

[40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014. Available: https://arxiv.org/abs/1406.1078

[41] S. Sarker, and M. M. Haque, "An approach towards domain knowledge based classification of driving maneuvers with LSTM network," presented at the 4th *Int. Conf. IJCACI,* Dhaka, Bangladesh, November, 2020.

[42] A. Mammeri, Y. Zhao, A. Boukerche, A. Siddiqui, and B. Pekilis, "Design of a semi-supervised learning strategy based on convolutional neural network for vehicle maneuver classification," in *Proc. WiSEE,* Ottawa, ON, Canada, 16-18 Oct. 2019, pp. 65-70.

[43] R. Stewart, and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge," in *Proc. IV,* vol. 31, no. 1, February, 2017.

[44] V. Kazak, "Unsupervised feature extraction with autoencoder: for the representation of parkinson´ s disease patients", Ph.D. dissertation, NOVA Information Management School, Universidade Nova de Lisboa, Lisbon, Portugal, 2019.

[45] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, "Deep learning," vol. 1, no. 2, Cambridge MIT press, 2016.

[46] D. Smirnov, E. M. Nguifo, "Time series classification with recurrent neural networks," *Advanced Analytics and Learning on Temporal Data,* vol. 8, 2018.

[47] Z. Wang, W. Yan, T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. IJCNN,* USA, 2017, pp. 1578-1585.

[48] A. Gulli, and S. Pal, "Deep learning with Keras," in *Packt Publishing Ltd,* 2017.

[49] Driver Behavior Dataset. [Online] Available: https://github.com/jair-jr/driverBehaviorDataset