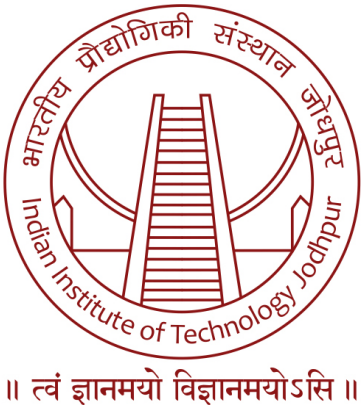
Hypergraph Based Optimal Route Identification with Driver Behavior Prediction as Beyond the Root Solution

A Project Report Submitted by

Nishit Bhardwaj

in partial fulfillment of the requirements for the award of the degree of

Master of Technology



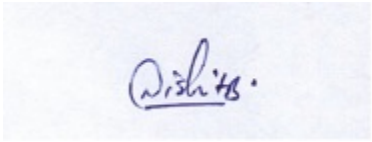
Indian Institute of Technology Jodhpur

Department of Computer Science and Engineering

May, 2022

Declaration

I hereby declare that the work presented in this Project Report titled Hypergraph Based Optimal Route Identification with Driver Behavior Prediction as Beyond the Root Solution submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of Master of Technology, is a bonafide record of the research work carried out under the supervision of Dr. Debasis Das. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.



Signature

Nishit Bhardwaj

M20CS067

Certificate

This is to certify that the Project Report titled Title of the Project Report, submitted by Nishit Bhardwaj (M20CS067) to the Indian Institute of Technology Jodhpur for the award of the degree of Master of Technology, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Dr. Debasis Das

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Debasis Das for enlightening me the first glance of research, for his constant impartment of knowledge, and for showing confidence in me.

I would like to acknowledge the technical support and help of Mr. Ankur Nahar, Ms. Bhumika, and Mr. Jayant Vyas.

Finally, my warm and heartfelt thanks go to my family for the tremendous support and hope they had given to me. I would like to dedicate my hard work to my late mother, Smt. Suman Sharma and my late brother, Aditya Bhardwaj.

Abstract

In this project work, our focus is to provide a network layer and application layer solution to better intelligent transportation systems. The application layer solution involves driver behavior classification, and the network layer solution consists in finding the optimal route path with proper interpretable decisions.

Improving driving safety by monitoring driver behavior is an excellent example of Advanced Driver Assistance Systems (ADAS). This project work proposes an end-to-end transformer-based driver behavior classification framework named TransDBC. It calculates driver behavior from the multivariate time-series smartphone telematics data by learning short and long-range temporal dependencies effectively and accurately, unlike prior data-driven deep learning models that capture information locally via convolutional or recurrent structure in an iterative manner. The extensive human-in-the-loop study on a publicly available UAH-DriveSet dataset shows that the proposed technique can classify unsafe driving behavior with a 96% of average weighted precision, recall, F-measure, and 95.38% accuracy. Our suggested model outperforms baselines and state-of-the-art models on the UAH-DriveSet dataset and five different multivariate time-series datasets in driving behavior analysis.

In recent years, the common trend shows the integration of deep learning frameworks with software-defined networking (SDN) as a key practice for the advancement of intelligent transportation services (ITS). Despite the remarkable performance of the deep learning-based networking systems, the lack of interpretability and heavy deployment features of deep learning models makes them non-desirable to highly dynamic, time and resource constraint scenarios such as vehicular ad-hoc networks (VANETs). In this project work, a deep deterministic policy gradient (DDPG) framework with the convolutional nueral network (CNN)- gated-recurrent units (GRU) is implemented to provide better interpretability and then convert into hypergraph for lightweight deployment. As the hypergraph contains only a few parameters compared to deep neural networks (DNNs), it outperforms the DNNs in terms of performance based on computation time, and interpretability parameters without affecting the better route selection decision of the system. Then the results obtained by deploying the DNN based model and the hypergraph-based model in the SDN routing system is compared.

Contents

List of Figures

List of Tables

Hypergraph Based Optimal Route Identification with Driver Behavior Prediction as Beyond the Root Solution

1 Introduction and background

The intelligent transportation system (ITS) goal is to provide smart and innovative solutions to traffic management problems to facilitate the user's more comfortable and safer experience. In this project work, we aim to bring ITS closer to achieving its goal. For the same, we have proposed a root or network layer solution as "A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability" and beyond the root or application layer solution as "TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification." The description of the two solutions is in the following sections.

1.1 TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification

Driving can be dangerous and needs monitoring to ensure the passenger, driver, and vehicle's safety. According to the World Health Organization's (WHO) Global Status Report on Road Safety, the number of yearly road traffic deaths has risen to 1.35 million [1]. Road accident trends in South-East Asia are increasing every year. In India, only 4,49,002 road accidents were recorded in 2019. Besides, as many as 50 million people are injured in road accidents [2]. One of the reasons for road accidents is the significant discrepancy between the transportation sector's growth and the availability of systems to make a safe road drive for passengers and drivers. This project work proposes a road traffic safety system based on driver behavior analysis. Driver behavior analysis research has been conducted for several years [3–7], to identify the human mistakes in driving which are governed by different parameters, such as tiredness, drowsiness, aggressiveness. We can measure these parameters by observing the driving patterns; for example, side slipping on the road may be caused by drowsiness. In this regard, we develop an intelligent system that classifies different driving behavior using smartphone telematics data collected from different inertial sensors (such as accelerometer, a global positioning system (GPS), gyroscope, altimeter, ambient light sensor).

Data collected by sensors contains multiple variables at each timestamp, called multivariate time-series (MTS), and is widely used in various fields, including astronomy, biology, geoscience, smart cities, health care, human action recognition, and other scientific and social areas. The essential challenge in MTS is multivariate time-series classification (MTSC). MTSC has gotten far less attention from researchers than univariate time-series classification (UTSC) because of its complex nature due to interaction dependency between variables. UTSC algorithms are used in MTSC problems by ensemble the result of applying UTSC on each channel independently [8].

A wide range of modeling approaches is available for univariate and multivariate time series. However, in classification tasks, deep learning models still face challenges in providing the state-of-the-art [9–11]. Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) are the most widely used deep learning-based methods for MTSC. There are two fundamental flaws in these models; first, these models are unable to capture and leverage long-term dependencies efficiently. Second, there is limited research on the interactions of multiple variables. CNN-based models combine local neighborhood data to build features, and RNN-based models use hidden states from past time steps to calculate features at each step. Experiments demonstrate that both CNN and RNN-based techniques have issues capturing dependencies to very long historical data. Late CNN layers in deep CNN models or RNN-based approaches can capture long-term memories.

Furthermore, training such deep neural networks is computationally inefficient. We propose a novel transformer-based model TransDBC for multivariate time-series classification to address the above-mentioned limitations. Transformers are a relatively new type of deep learning model that was initially introduced for natural language translation [12] but have since monopolized state-of-the-art performance across practically all NLP [13] and vision tasks [14]. The multi-headed attention mechanism of transformer models makes them ideal for time-series data. Attention heads can consider multiple aspects of relevance between input elements for time series, i.e., different representation subspaces, corresponding to multiple periodicities in the signal; this may correspond to multiple periods in the signal. To take advantage of multivariate time-series data for classification tasks, we create a TransDBC based on the impressive results obtained by transformer models in NLP and vision tasks.

The challenge of short and long-term dependency in ADAS we tackled through the proposed model, as follows: A value at time step t may be influenced by its immediate historical values or by values that occurred far before time stamp t. For example, the aggressive driving behavior may be observed by the historical driving patterns where the driver may cut off other drivers in traffic by swerving between lanes which is a reckless and dangerous act. Another challenge we tackled is considering the relationships between multiple variables such as acceleration, latitude, longitude, and gyroscope for the behavior analysis.

Our main contributions are as follows:

- The use of a novel transformer-based TransDBC model can capture short and long-term dependencies to accurately classify driver behavior into normal, drowsy, and aggressive.
- The extensive experiments on the UAH-DriveSet dataset show the performance of TransDBC in terms of accuracy, precision, recall, and F-measure metrics is better than existing baseline and state-of-the-art models.
- We conducted the experiments on five different multivariate time-series datasets and observed that the proposed architecture outperforms existing state-of-the-art models and shows the effectiveness of the proposed model over other MTSC tasks.

1.2 A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability

Software-defined networking (SDN) provides efficient message propagation in a Vehicular ad-hoc Networks (VANETs) environment. SDN exploits the central controller-based architecture by gathering the network information from the local nodes and compute the optimal route [15]. However, the SDN-based architecture faces several challenges such as real-time message dissemination, network congestion, and highly dynamic topology because of traffic instability. The superior performance of the deep neural networks (DNNs) surges the capabilities of deep learning applications across a wide range networking systems and also perform network route optimization, such as SDN routing optimization [16], Video streaming [17], and On-switch flow scheduling [18]. The ability of DNNs to get integrated with the optimization technique such as reinforcement learning is the primary reason behind the success of deep learning in networking systems.

However, despite the superior performance of DNNs, the network operators are reluctant to use deep learning practically. The reasons behind this reluctance are: 1) the use of a large number of free parameters in the DNN models. 2) the DNN model is considered as a black-box and is uninterpretable for the network operators. A large number of free parameters inside the DNN model makes it difficult to use in resource constraints and real-time environments due to the bulky size and response time required respectively. Also, due to the structural design the DNNs are considered as blackboxes and it is quite difficult to interpret them and therefore DNNs are difficult to debug and ad-hoc adjust [19]. This project work involves hypergraph and deep learning-based deep deterministic policy gradient (DDPG) technique implementation using an actor-critic architecture that can operate over continuous action spaces in VANETs environment [20]. The traffic metrics generated by SDN contain spatio-temporal data because of the spatial correlation and temporal variation of traffic flows [21]. To address the Spatio-temporal data, a combination of convolutional neural networks (CNN) and gated-recurrent unit (GRU) is adopted. CNN is widely used to extract spatial features but it can not capture the temporal variations. Furthermore, the temporal feature extraction is addressed by GRU. Once the CNN+GRU model is trained, it is used to formulate a hypergraph [19]. The obtained hypergraph is then used to find the optimal route path. The main advantage of deploying a hypergraph instead of DNNs is that the hypergraph contains few parameters and therefore it is lightweight to deploy, quick response, easy to interpret, and ad-hoc adjust.

The proposed approach consists of an SDN server communicating through roadside units (RSUs). The communications between RSUs and the vehicles are based on dedicated-short range technology (DSRC) using the 802.11p standard. The RSUs can be found in the different sections of the road. The in-range vehicles can communicate with each other, while in the case of the non-availability of other vehicles, they can communicate through RSUs. SDN is directly connected to the RSUs through an IP network. The SDN control plane is responsible for the policy exchange and creating new routing policies. RSUs are used to communicate with SDN using a hybrid scenario.

2 Literature survey

2.1 TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification

This section briefly reviews two lines of research works closely related to our studied problem. In particular, we first discuss how the existing systems dealt with driver behavior analysis. Then, we introduce the work which utilizes deep learning-based methods for multivariate time-series classification (MTSC).

2.1.1 Driver Behavior Classification

As ADAS emphasizes, the classification of driving style is becoming increasingly important in terms of safety and energy efficiency. Pjetri et al. [22] presented a classification method using the minimum number of features to reduce the computing overhead, and the random forest is used for the classification of driver behavior into two categories aggressive and normal driving. Carvalho et al. [23] used the Recurrent Neural Network (RNN) for driving style detection. They took advantage of the internal memory of RNN, where it stores information about the past, which helps update the hidden states. The dataset used has only 69 samples collected from smartphones sensors, which is not suitable to evaluate this type of road safety system. It detects only aggressive driving style out of three. Khodairy et al. [24] presented a driver behavior classification method on oversampled signals of sensors embedded on a smartphone. They used LSTM neural network in a stacked manner for optimized classification. Kouchak et al. [25] presented a bidirectional LSTM network model and attention network model for driver behavior estimation. They identified the driver's behavior, which is causing a distraction to his driving. Moukafih et al. [26] presented a fully convolutional network-based LSTM model (LSTM-FCN) for driver behavior classification. In FCN for feature extraction, various convolutional layers are used. The second part is the recurrent neural network-based LSTM model, which handles the problem of vanishing gradient. Compared to other techniques, the proposed methods perform better in terms of processing time window size. However, the proposed method's performance degrades with increasing window size (i.e., > 5 minutes).

2.1.2 Deep Learning For MTSC

Sensor technology advancements have made the multivariate time-series classification (MTSC) problem, one of the most important in the time-series data mining domain, a major focus in recent years. An attentional prototype network is proposed in [27] to take advantage of both traditional and deep learning-based approaches in the MTSC model. Random group permutation and multi-layer convolutional networks were used to discover low-dimensional features from multivariate time series data. A novel attentional prototype network is proposed to deal with the problem of limited training labels. The feature representation is trained based on its distance to class prototypes with inadequate data labels. Multivariate LSTM-FCN (MLSTM-FCN) [11] has shown successful to classify multivariate time-series. The model can learn combined features by performing convolution procedures to all variables; however, the combined features ignore pair-wise dependencies between two variables.

Furthermore, these models are unable to represent long-term dependencies efficiently and effectively. For MTSC, the author in [28] proposed a Multi-Channel Deep Convolutional Neural Network (MC-DCNN). MC-DCNN accepts input from each variable to find latent features, and the latent features from each channel are put into an MLP for classification. In another work, [29] to pre-process the data, a multi-scale convolutional neural network (MCNN) employs down-sampling, skip sampling, and a sliding window. The MCNN classifier's success is heavily dependent on the dataset's pre-processing and the tweaking of a large number of model hyperparameters.

2.2 A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability

In this section, we have mentioned the related work in three subsections: deep learning model for routing optimization, SDN in VANET, and interpretability in deep learning-based models. The problem statement is to obtain a high-performing network, better resource utilization, low computation overhead, adaptable, economical, and easily deployable system by observing traffic metrics and finding corresponding spatial and temporal variations to select the best path to the destination. Furthermore, It is required to obtain a system that is easily interpretable and flexible to ad-hoc adjust.

2.2.1 Deep learning in routing

In recent years, many researchers have used deep learning in networking systems to perform various network optimization tasks. Specifically, in routing, various systems provided deep learning-based solutions as given in the surveys [30, 31], they mentioned SDN applied machine learning algorithms in different tasks like routing optimization, traffic classification, QoS prediction, etc. In [32], used deep learning-based techniques for the software-defined communication systems. They used an online training paradigm for CNNs that compute paths based on real-time traffic traces. The authors in [33] provided a routing framework that used deep learning models trained on traffic flow and evaluated the problem of network congestion metrics on real-world datasets. Another work given by authors [32] used supervised deep learning systems for constructing routing tables and provided simulation results in terms of throughput, delay, and signaling overhead. The use of deep learning-based approaches are widely adopted in solving routing problems in transportation networks also such as in [34], the authors provide the solution to routing problems in intelligent transportation systems using pointer networks and transformer attention networks. Another study [35] used deep reinforcement learning for monitoring the movement of the vehicle and creating a suitable path for routing packets in a VANET environment. The simulation results show the effectiveness of routing packets between a source and a destination vehicle. An intersection-based Q-learning protocol for V2X has been introduced in [36]. The authors used a multimodal Q-table and a greedy forwarding technique to decrease the end-to-end delay and improve the packet delivery ratio. However, the authors omit the orientational movement of the vehicles apart from intersections,

and resource management in a multidimensional model is an open challenge. A Q-learning-based routing protocol using SDN has been introduced in [37] to improve packet delivery ratio and transmission delay. The authors have implemented a GMM based classifier along with Q-learning to handle packet forwarding decisions. However, the proposed method used a distributed environment and increased control packet transmission in the network. Moreover, Q-learning-based routing failed to address the continuous nature of VANETs efficiently. A DRL-based SDN architecture has been utilized in [38] to improve the convergence and throughput of the system. The authors used the DDPG technique to optimize paths and reward calculation in SDN. The DRL-based agents used network delay and traffic matrices to improve the effectiveness of the scheme. However, the technique suffers from the overfitting of correlated data in some scenarios. A link continuation model using a time duration and link duration has been proposed in [39]; the authors use link stability as a key parameter to improve network performance. However, the authors do not consider the arbitrary movement of the vehicle, which leads to differing probabilities in state transitions. ML and artificial intelligence techniques have been utilized in [40] to predict mobility and enhance routing performance. However, the scheme suffers from frequent disconnection scenarios due to vehicle movement and geographical information requirements.

2.2.2 SDN based solutions in VANETs

SDN is also gaining popularity in various networking solutions. A graph-theory-based multi-flow congestion-aware routing algorithm has been proposed in [41] to select an optimal path based on the path length and the congestion level. A Game-theory based SDN architecture has been proposed in [42] to choose the most feasible network. To support QoS requirements, in [43] authors used the link stability and latency as the primary parameters to find the optimal shortest path. A social-aware clustering with semi Markov model [44], and SDN along with ML [45] were also used to find the optimal path and efficient traffic analysis prediction. A delay-constrained routing strategy in [46] discussed the problems related to scattered traffic conditions in software-defined bus ad-hoc networks. A novel architecture integrating 5G-based SDN in ITS has been proposed in [47]. However, their approach has the drawback of a shorter range in 5G technology, which delivers less scalable and feasible networks.

2.2.3 Interpretability in deep learning-based models

To interpret deep learning-based models is a hot topic in the machine learning (ML) community. Many researchers have given different approaches to interpreting DNNs, such as CNN [48], RNN [49], and Long short-term memory (LSTM) networks [50]. The interpretation methods depend upon the application domain, such as visual recognition [48], natural language processing [51], and speech recognition [52]. For our approach, a suitable CNN and GRU interpretation mechanism in the networking system domain is required. For networking systems, DNNs can be interpreted by either converting the model into a decision tree or by formulating a hypergraph. The networking systems are classified into two classes: local systems and global systems. The local system is the system that collects information locally and makes decisions based on the collected information, while the global system is the system that collects

information across the network and makes decisions according to global planning. The local systems are interpreted by converting them into decision trees, where the interpretation is obtained by analyzing the decision-making nodes in the decision tree and input data point. The global systems are interpreted by formulating a hypergraph as the hypergraph structure is most suitable to interpret a decision based on global parameters such as topology and input in global networking systems are graph-structured [19].

3 Problem definition and Objective

3.1 TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification

In this section, we first introduce some key preliminary definitions before outlining the overall purpose of our research.

Definition 1 (Multivariate Time-series)**.** In a multivariate time series, more than one time-dependent variable is involved, and each variable is dependent on the previous values of the other variables. We consider multivariate time-series as data collected from different smartphone sensors represented as $X = \{x_1, x_2, \ldots, x_m\} \in R^{m \times w}$, where m is the dimension and w is the length of series.

Definition 2 (Driver Behavior)**.** Driver behavior describes the characteristics and actions while operating a vehicle. We consider aggressive, drowsy, and normal behavior as A_{gg}, D_{dw}, and N_{ml} for n number of drivers.

Problem Statement: We formulate the problem statement as Let each multivariate time-series have a class label $y \in \Psi$ from a predefined label class Ψ. In the case of driver behavior classification, let D_i represents a driver where $i = 1, 2, ..., n$ be the driver identity and B_i denotes the behavior of a driver. Given the historical data $X = \{x_1, x_2, \ldots, x_9\} \in R^{9 \times 64}$ with Ψ behavior categories $\Psi = (A_{gg}, N_{ml}, D_{sy})$ and group of multivariate time-series $\Lambda = \{X_1, X_2, \ldots, X_k\} \in R^{k \times m \times w}$, where k is the number of time-series of n drivers and A_{gg}, D_{dw}, and N_{ml} denotes the aggressive, drowsy, and normal behavior. The objective of TransDBC is to learn a classifier to learn predictive function that infer driver behavior of each category in future time steps. The proposed model predict the behavior B_{i+1} using function f represented by following equation:

$$O_w^{c,(k+s)} = f(\mathcal{GPS}, \mathcal{ACC}, \mathcal{GRY}, \mathcal{DST}) \tag{1}$$

Where mapping function $f(.)$, we want to learn from GPS data ($\mathcal{GPS}$), accelorometer ($\mathcal{ACC}$), gyroscope ($\mathcal{GRY}$), and distance ($\mathcal{DST}$) as an argument. The output is $O_n^{c,(k+s)}$ which represents the prediction of behavior category in s future time steps.

3.2 A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability

The proposed approach is motivated by the following study:

- The deep learning models are heavy to deploy and require more computation time which is not suitable for VANET environments and the lack of interpretability of the deep learning model is also an issue to address.

- Conventional routing algorithms cannot be used to optimize frequent route discoveries without considering the relationships between the subgraphs, creating the motivation of our research.

Most of the learning techniques use experience relay and have a slow learning rate. The continuous action space of VANET demands a model-free route selection technique that can work on continuous value instead of discrete action space. Link quality and transmission efficiency is also the primary requirement for improving network performance. It is necessary to include QoS parameters to ensure highly stable and robust network performance. Furthermore, deploying the heavyweight DNN models such as CNN and GRU in a resource constraint, real-time computation, and real-time communication-based system is inefficient in terms of computation time and interpretability. There are many times when it is required to provide ad-hoc adjustments in the deployed model. However, in the case of the CNN-GRU model, it is challenging to achieve this since they contain millions of parameters and have incomprehensible structures. Therefore, to resolve the aforementioned issues, the objective is to obtain:

- a deep learning-based solution for route optimization using the features such as the time dependency, routing cost, velocity, and traffic load vehicular scenario.
- an appropriate interpretation method to back the deep learning-based networking model .
- a lightweight solution that can be deployed in a resource constraint environment.

4 Methodology

4.1 TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification

The proposed methodology comprises four steps: 1) Data pre-processing, 2) Feature selection and extraction, 3) Transformer based model architecture, and 4) Results analysis. Details of these steps are provided in the following sections.

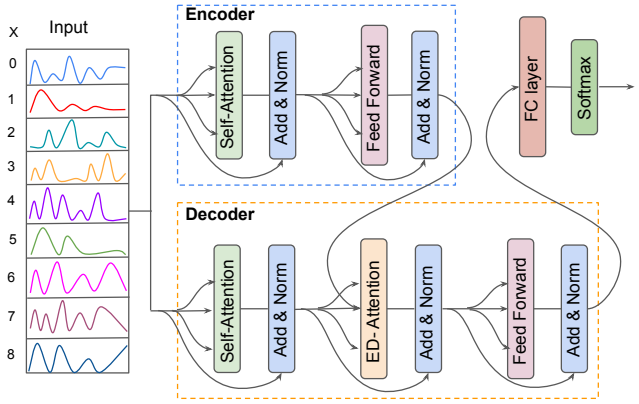


Figure 4.1: Transformer Model for Multivariate Time-Series Classification

4.1.1 Data Pre-processing

Dataset is first pre-processed before being fed into the model for driver behavior classification. Pre-processing is required due to the collection of the dataset on different frequencies between 1 Hertz to 100 Hertz by various sources such as accelerometer, gyroscope, GPS. k-nearest neighbors (k-NN) is used to pre-process the dataset to remove the duplicate rows and normalize the data to suit the model better. The whole data is synchronized on the 10 Hertz frequency. Data is stored in different files based on the driver behavior and road types. A sliding window is used to divide the data into overlapping segments. Data enhancement is also applied to improve the model performance. Finally, normalization of the dataset is carried out to scale the dataset between 0 to 1 using the following formula:

$$Feature_i = \frac{Feature_i - min(Feature_i)}{max(Feature_i) - min(Feature_i)} \tag{2}$$

where $Feature_i$ denotes the i^{th} feature and min, max denotes the minimum and maximum value of that feature in the feature set.

4.1.2 Feature Selection and Extraction

Feature selection and extraction are essential tasks in any machine learning problem to make the best fit of the dataset on the model. It is a technique to find the relevant features in the dataset by minimizing the loss of information by removing the other features from the dataset. In our case, we used the recursive feature elimination method. Cross-validation is used to select the best features based on the accuracy metric. We applied thirteen statistical computations on the selected feature: max, min, mean, median, standard deviation, kurtosis, skewness. Finally, we got 208 features for the experiments. On these 208 features, we applied the feature selection method to get the best feature set that contributes most to deciding the driver behavior.

4.1.3 Transformer Model Architecture

Most of the researchers used the transformer model for classification tasks in recent trends [53–55]. We switched from the image and text to multivariate time-series data-based driver behavior classification. The overall architecture of the Transformer model is shown in Figure 4.1

Encoder-Decoder The proposed transformer model is based on the original transformer work by Vaswani et al. [12]. Here, the encoder is given an input samples of multivariate time-series data $x_i \in R^m : X \in R^{m \times w} = (x_1, ..., x_w)$, true labels $Y = (y_1, ..., y_m$ and it results a sequence of continuous representations $Z = (z_1, ..., z_m)$. Given Z and the same x_i fed into the decoder which generates an output sequence $O = (o_1, ..., o_m)$. The model is auto-regressive, consuming the previously generated sequence as additional input when developing the next sequence. Encoder and decoder are multi-layer stacks, and when we increased the layers up to six, as shown in the Figure 5.3, where it gives the maximum F-measure score. All encoders are identical in structure, which is applicable for all decoders. The input-output connection and workflow of encoder and decoder layers are described in Algorithm 1 from lines 2 to 7.

A single encoder layer consists of a self-attention layer and a feed-forward layer sequentially, enabled with residual connections at every layer. The residual connections are formed with the help of the "Add & Norm" Layer, which provides addition and normalization operations. The encoder's inputs first pass through a self-attention layer, which allows it to keep track of other data in the input sequence X while encoding a single piece of information. The output of the self-attention layer is added with the initial input and further normalized through the Add & Norm layer, as shown in the following equations.

$$A = multiHeadAtt(in, N_{head}) \tag{3}$$

$$A^{'} = Norm(A + in) \tag{4}$$

Here, in is the initial input to the encoder layer, N_{head} denotes the number of parallel attention layers for self-attention layer, A, and $A^{'}$ denotes the output of the self-attention layer and Add & Norm layer

respectively.

A feed-forward neural network processes the output of the Add & Norm layer. Each layer of the encoder has the same feed-forward network applied to it. The output from the feed-forward network is again fed to the second Add & Norm layer to form a second residual connection inside a single encoder layer as shown in the Equation 7 and 8.

$$f = FF(A^{'}, ff_{dim}) \tag{5}$$

$$f^{'} = Norm(f + A^{'}) \tag{6}$$

Here, ff_{dim} represents the dimension of the feed-forward neural network, f and $f^{'}$ represents the output of the feed-forward neural network and Add & Norm layer, respectively. The overall step-by-step working is embedded in Algorithm 1 from lines 8 to 14. The total number of encoder layers in the encoder block is denoted by N_e.

The structure of a single decoder layer is very much similar to the encoder layer. The difference between the encoder and decoder layers is that a decoder layer consists of extra self-attention and Add & Norm layers. These additional layers take input from the output of the encoder block and the output from the previous Add & Norm layer as shown in Figure 4.1. The equations involved in the steps mentioned above are described as follows:

$$A_1 = multiHeadAtt(A^{'}||en_{out}, N_{head}) \tag{7}$$

$$A_1^{'} = Norm(A_1 + A^{'}) \tag{8}$$

where en_{out} denotes the output of the encoder block, A_1, and $A_1^{'}$ denotes the output of the self attention and Add & Norm layer, respectively. The total number of decoder layers in the decoder block is represented by the symbol, N_d. The overall decoder block is mentioned in the Algorithm 1 from line 15 to 23.

The length of each sample in a time series is an important consideration to keep in mind. Using a padding mask, which adds large negative values to the attention scores for the padded positions before calculating the self-attention distribution with a softmax function, shorter samples in the dataset are padded to the maximum length w allowed for the dataset. As a result, the model is forced to ignore padded positions while processing large mini-batches of samples simultaneously.

The Model Objective The model objective/loss function is the one that needs to be optimized. In our case $L\big(Y_j, O_j\big)$ is cross entropy loss because we are dealing with classification problem.

$$L\big(Y^j, O^j\big) = -\sum_{\Psi=1}^{3} Y_{j,\Psi} log(O_{j,\Psi}) \tag{9}$$

Algorithm 1: The learning process of TransDBC

Input: X and Y
Output: O
Paras: D, N_{head}, N_e, N_d, b_{size}, ff_{dim}, and lr

Initialize all parameters
// Sample a minibatch of size b_{size}
foreach $(X_B, Y_B) = sample(X, Y, b_{size})$ **do**
  foreach $(X_B^m, Y_B^m) \in (X_B, Y_B)$ **do**
    $Z_B^m = encoder(X_B^m, N_e, ff_{dim}, N_{head}, D)$
    $O_B^m = decoder(X_B^m, Z_B^m, N_d, ff_{dim}, N_{head}, D)$
    Update loss L as $L = Lossfn(O_B^m, Y_B^m)$
  Update all parameters w.r.t. L;
Function encoder(in, N_e, ff_{dim}, N_{head}, $dropout$):
  foreach $n \in N_e$ **do**
    $A = multiHeadAtt(in, N_{head})$
    $A^{'} = Norm(A + in)$
    $f = FF(A^{'}, ff_{dim})$
    $f^{'} = Norm(f + A^{'})$
  return $f^{'}$
Function decoder(in, en_{out}, N_d, ff_{dim}, N_{head}, $dropout$):
  foreach $n \in N_d$ **do**
    $A = multiHeadAtt(in, N_{head})$
    $A^{'} = Norm(A + in)$
    $A_1 = multiHeadAtt(A^{'}||en_{out}, N_{head})$
    $A_1^{'} = Norm(A_1 + A^{'})$
    $f = FF(A_1^{'}, ff_{dim})$
    $f^{'} = Norm(f + A_1^{'})$
  return $f^{'}$

where $Y_{j,\Psi}$ and $O_{j,\Psi}$ are the ground truth and predicted score of j is of class Ψ i.e., (A_{gg}, D_{rw}, N_{ml}).

4.2 A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability

Our proposed approach uses SDN and OpenFlow standards for routing decision-making and policy exchange purposes. The SDN architecture has been configured according to the settings of VANETs. The packets are organized as the flows, and each flow contains multiple data packets. Data flow direction and the information is contained in the flow tables, and the flow entries specify the action to be executed. In our proposed scheme, the agents i.e., the Actor-Critic network based on the CNN-GRU model generate the training data.

4.3 Deep Learning Model For Route Optimization

The proposed approach follows the implementation of the DDPG model for route optimization. The DDPG model can efficiently address the continuous action space. The beacon messages are passed by

the vehicle and the communication nodes in the vehicular network to obtain information such as vehicle velocity and geographical positions. The RSUs collect the relevant information in the network, which is then passed to the SDN. The SDN provides computing and communicating the efficient path from the source to destination to the involved network nodes. The network information consists of the source node, destination, bandwidth, the distance between the nodes, and link availability as parameters.

The DDPG model is created by defining the state, action, and reward. The agent inputs the environmental parameters and produces the action as output. In our approach, the vehicle is considered the agent, and its neighbor vehicle is considered the environment. The action represents a path from source to destination. The reward function is designed to provide a position reward when an efficient path is selected and a negative reward otherwise. The reward function utilizes distance between the pair of vehicles, link delay, and link availability as performance parameters to provide rewards.

After getting the traffic matrix, the CNN-GRU-based actor-critic model is trained according to the policy gradient algorithm. The CNN-GRU model can efficiently extract the Spatio-temporal features from the traffic matrices. In the proposed approach, both actor and critic use the same CNN-GRU architecture in which the CNN and GRU models are combined in parallel connection. The actor takes traffic matrix as input and produces action as output while the critic takes both action and traffic matrix as input and produces a discount factor and reward-based value function as output. The CNN layer contains 5 layers which are convolution, batch normalization, Max-pooling, dropout, and attention layer as shown in the figure4.2. The CNN model can efficiently extract spatial features. The CNN extracted features are parallelly fed into the GRU model to extract the temporal features. The GRU block used in the proposed approach is shown in the figure4.3.

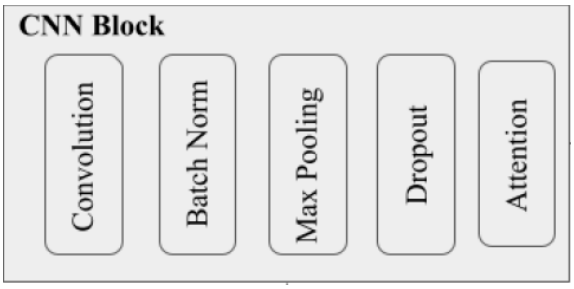


Figure 4.2: CNN Block

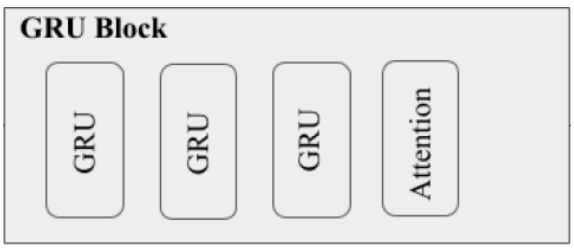


Figure 4.3: GRU Block

4.4 Hypergraph Formulation

A hypergraph is a generalization of a graph that consists of vertices and hyperedges. A hyperedge can connect any number of vertices. An example of a hypergraph is shown in Fig 4.4.

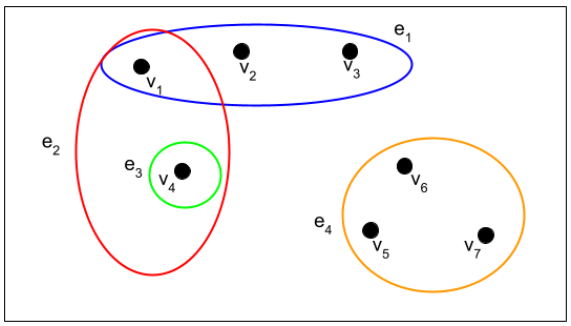


Figure 4.4: Hypergraph Example

Let the hypergraph be represented as H then, H is a pair i.e. $H = (V, E)$ where V is the set of vertices and E is a set of non-empty subsets of V called hyperedges. The vertices set V is identified as $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. The shown hypergraph contains four hyperedges and the set E is represented as $E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_1, v_4\}, \{v_4\}, \{v_5, v_6, v_7\}\}$. A hypergraph can be represented by an incidence matrix with dimensions $\mid E \mid \times \mid V \mid$. The incidence matrix is a binary matrix that contains the relationship between vertices and hyperedges. The incidence matrix for the example in Fig. 4.4 is:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{10}$$

Where, $M[i][j] = 1$; if i^{th} hyperedge contains j^{th} vertex.

According to the description of the hypergraph, it consists of vertices V and hyperedges E. In our approach, the SDN optimizer selects the most efficient route from source to destination based on the output of the deep learning model. This path selection depends on the topological data such as link availability, link delays and link capacity. Such types of systems are suitable for formulating a hypergraph where the hyperedges represent the route path from source to destination, and the vertices represent the communication links between two nodes in the network.

Hypergraphs can efficiently represent path information while maintaining each vertex v and hyperedge e related features. The hyperedge features are traffic demand volume between each source and destination pair, and the vertex features are the link capacity. Fig. 4.5 demonstrates a hypergraph representation for a given topology. The simplified graph structure of the topology is shown in Fig. 4.5(b). The links numbered from 1 to 8 act as vertices of the hypergraph. Let the routing path obtained using the trained model between nodes RSU R_0 as source and vehicle V_4 as the destination is $\{1-3-7-8\}$ and between nodes vehicle V_5 and RSU R_0 is $\{9-6-5\}$. The hyperedges e_1 and e_2 in

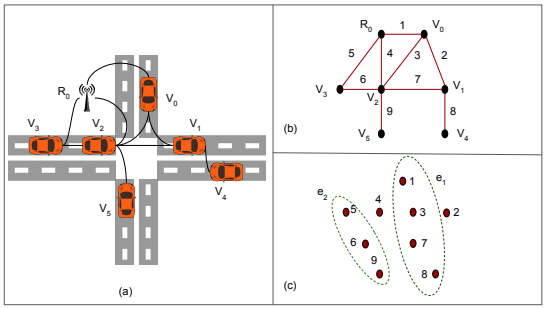


Figure 4.5: The hypergraph representation of the SDN routing model

the hypergraph represent the paths mentioned above, and therefore the connections in the hypergraph are: $\{(1, e1), (3, e1), (7, e1), (8, e1), (5, e2), (6, e2), (9, e2)\}$.

The interpretability in the hypergraph is obtained by implementing the critical connection search method [19]. This method evaluates the criticality of a link present between two nodes in a path selection decision represented by the mask value of the link. The mask value is obtained by optimizing parameters related to the performance degradation, interpretation conciseness, and determinism. The performance degradation parameter ensures higher weight to the link having too much influence on the routing decision. The interpretation conciseness parameter is related to the total number of critical connections present in the whole topology. The determinism parameter is included to obtain the value of most critical connections close to one and least critical connections close to zero.

5 Theoretical/Numerical/Experimental findings

5.1 TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification

In this section, the experimental setting that we use in our experiments, such as datasets, metrics, baselines, and state-of-the-art models, are described at the beginning. Then the results analysis part is covered, which is comprised of comparisons and hyper-parameter settings.

We aim to answer the following research questions in upcoming sections:

- **RQ1** How does TransDBC performs in terms of different accuracy metrics?
- **RQ2** How does our framework works as compared to baseline techniques in classifying driver behavior?
- **RQ3** How does our model works as compared to state-of-the-art techniques?
- **RQ4** What is the performance of TransDBC on other multivariate time-series datasets?
- **RQ5** How does the performance varies with different hyperparameter setting?

Driver	***Genre***	***Age range***	***Vehicle Model***	***Fuel type***
D 1	Male	40-50	Citroen C-Zero (2011)	Electric
D 2	Male	30-40	Opel Astra (2007)	Gasoline
D 3	Male	20-30	Citroen C4 (2015)	Diesel
D 4	Male	40-50	Audi Q5 (2014)	Diesel
D 5	Female	30-40	Kia Picanto (2004)	Gasoline
D 6	Male	20-30	Mercedes B180 (2013)	Diesel

Table 5.1: Description of Drivers and Vehicles participated in UAH-DriveSet Data Collection

5.1.1 Experiential Settings

Dataset UAH-DriveSet [56] dataset is used for evaluating the model performance. Dataset collection was done by driving the monitoring app DriveSafe. Total 6 drivers participated in this dataset collection, out of which 5 were male, and 1 was female between the ages 20-50 years shown in Table 5.1. The Dataset is labeled in three behaviors (aggressive, drowsy, and normal). Two different types of road (secondary road and motorway) were taken for the drive. Motorway road was with 2, and 4 lanes in each direction with maximum allowed speed 120 km/h whereas secondary road was with 1 lane in each direction with a maximum allowed speed of 90 km/h. On motorway road type, each driver performed a total 3 trips (round-trip) of around 25km each, and on secondary road type, four trips (one-way) were performed

by each driver of around 16km each. The total dataset is 500 minutes of naturalistic driving. Three different fuel vehicles (diesel, gasoline, and electric) are used in the data collection. The dataset contains raw real-time data, processed data events, and continuous variables. It has features such as Timestamp, Acceleration in X, Y, and Z-axis, Roll, Pitch, Yaw, GPS Speed, Distance to ahead vehicle, Latitude, Longitude, Altitude. They also provided a DriveSet Reader, a tool to read all the data along with the captured video.

Another multivariate time-series datasets we used for our experiments are JapaneseVowels [57], ArabicDigits [58], KickvsPunch [59], CharacterTrajectories [60], and Wafer [59].

Experimental Setup The experiments are performed on Google Colab with CPU Intel Xeon 2.20 GHz and GPU specification Tesla P100-PCIE-16GB. Memory space allotted by Colab environment was 12 GB RAM and Hard disk space 34 GB. We implemented the model in Python using PyTorch toolbox. The hyper-parameters of the TranDBC model for the multiple experiments are initialized as follows: $N_e = 6$, $N_d = 6$, $N_{head} = 3$, $FF_{dim} = 32$, dropout $(D) = 0.1$, and batch size $(b_{size}) = 32$. Adam optimizer is used for training with learning rate of $(lr) = 0.001$ and run for 1100 epochs.

Performance Measures Performance measures are used to evaluate the model quantitatively. We use performance measures, namely accuracy, precision, recall, F-measure, accuracy, as metrics for evaluation as they are popular metrics for classification. Accuracy is defined as the ratio of the number of instances correctly predicted as normal, drowsy, and aggressive upon total instances of driving behavior given as $A_{acc} = \frac{P_r}{T}$. Here, A_{acc} represent the accuracy, P_r is the correctly predicted instance and T denotes total instances. F measure is calculated using the weighted harmonic mean of precision (P) and recall (R).

$$F = 2\frac{P * R}{P + R} \tag{11}$$

Here, P is the ratio of correctly classified relevant driver behavior and actual behavior. R is the ratio of correctly classified relevant driver behavior and predicted behavior.

In addition, we also calculated the macro average and the weighted average for precision, recall, and F-measure metric as a performance measure which is an important performance measure in the case of multi-class classification.

5.1.2 Evaluation

RQ1: Performance of TransDBC on UAH DriveSet Dataset We evaluate our model on the basis of different metrics and obtain accuracy 95.38%, macro recall 0.95%, weighted recall 0.96%, macro precision 0.96%, weighted precision 0.96% and macro F-measure 0.95%, and weighted F-measure 0.96%. The epochs versus accuracy graph is shown in Fig. 5.1, we obverse that accuracy increases with a number of epochs, and after 1100 epochs, the accuracy remains steady. Another evaluation is represented with the help of confusion matrix as shown in Fig. 5.2, where dark color shows the number of instances classified

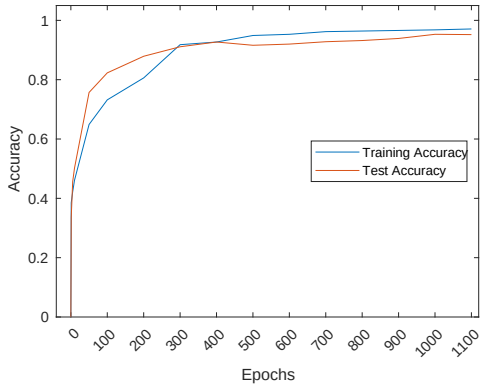


Figure 5.1: Accuracy v/s Epochs curve of Transformer Model

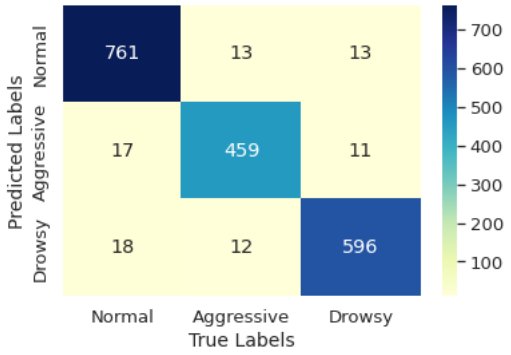


Figure 5.2: Confusion matrix of Classification Performance

correctly. We observe the dark color pattern in diagonal and light color pattern in the off-diagonal part, which means our model accurately classifies most of the instances.

RQ2: Comparison with Baseline Models We compare our TransDBC model with baseline deep learning models such as CNN, GRU, CNN-GRU, Transformer-Encoder (T-En), and Transformer-Encoder-GRU (T-En-G). The CNN network that we have used for comparison consists of 2 convolutional layers with a kernel size of 8×3. The GRU model consists of 2 GRU layers with a hidden layer size of 128. The CNN and GRU models described above are combined sequentially to obtain the CNN-GRU model. The T-En model is the transformer model after removing the decoder layers consists of only 6 encoder layers. The T-En model is combined with the GRU model to obtain the T-En-G model. The results of the above comparisons are described in Table 5.1.2. The CNN network captures inter-variable and temporal relations. However, it captures only local neighborhood patterns. Next, GRU can learn temporal relations more accurately but fails to learn long-term dependencies. CNN-GRU combined model provided 92.73% accuracy; still, there is a scope of improvement. At last, we try transformer-based models that capture long terms patterns along with inter-variable relationships. Among transformer-based models, TransDBC outperforms with an accuracy of 95.38% which is better than all recurrent neural network variants because of the ability of transformer-based architectures to work well for long sequences.

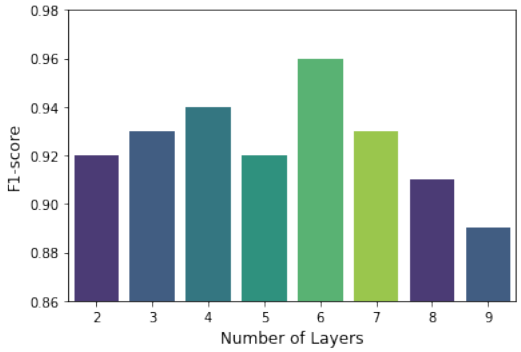


Figure 5.3: F-measure v/s Number of Layers in TransDBC

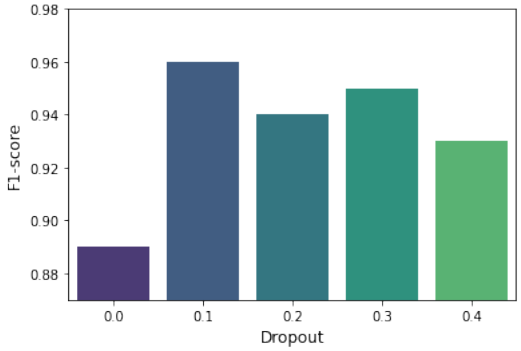


Figure 5.4: F-measure v/s Dropout Ratio in TransDBC

RQ3: Comparison with State-of-the-art Models We have compared our proposed TransDBC model with the existing state-of-the-art models for driver behavior prediction in terms of F-measure. Our proposed model outperforms state-of-the-art models. In [61] authors used raw smartphone sensor data as input to LSTM that classify driving behavior with F1 score as 0.91, and authors in [62] achieved an F1 score of 0.87 that used hand-crafted features with the random forest as classifying technique. [26] proposed novel Long Short Term Memory Fully Convolutional Network (LSTM-FCN) to classify aggressive behavior with an F1 score of 0.95 and [22] used random forest on the subset of features to classify behavior with 0.87 F1 scores. [63] applied random forest to personalize the driver state recognition system. TransDBC outperformed the state-of-the-art methods that used the UAH-DriveSet dataset. The model shows an accuracy of 95.38% in predicting driver behavior. In general, it is observed that the behavior predicted by LSTM is more regular than the TransDBC. Deep learning-based approaches work better than classical machine learning approaches as per the observation from Table 5.1.2.

RQ4: Performance of TransDBC on different Datasets Other multivariate time-series datasets we used for our experiments are JapaneseVowels [57], ArabicDigits [58], KickvsPunch [59], CharacterTrajectories [60], and Wafer [59]. Performance (accuracy in %) of different state-of-the-art models alongwith TransDBC on these datasets is described in Table 5.1.2. It can be observed that our model outperfroms GTN [64] on 3 out of 5 datasets showing an exceptional improvemnt from 90% to 100% on KickvsPunch [59] dataset.

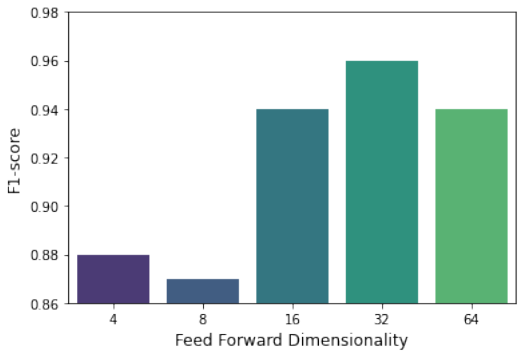


Figure 5.5: F-measure v/s Feed Forward Layer Dimensions in TransDBC

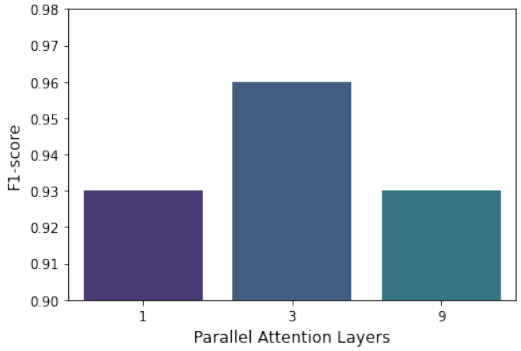


Figure 5.6: F-measure v/s Parallel Attention Layer in TransDBC

Authors	***Model Used***	***F-measure***
Saleh et al. [61]	Stacked-LSTM	0.91
Xie et al. [62]	Random Forest	0.87
Yi et al. [63]	Random Forest	0.91
Schlegel et al. [65]	HDC	0.94
Mekki et al. [66]	RNN/LSTM	0.90
Xie and Zhu [67]	Random Forest	0.88
TansDBC	Transformer	**0.96**

Table 5.2: Comparison with state-of-the-art models on Uah dataset

Model	**Precision (Avg.)**		**Recall (Avg.)**		**F1-score (Avg.)**		**Accuracy (%)**
	Macro	**Weighted**	**Macro**	**Weighted**	**Macro**	**Weighted**	
CNN	0.82	0.81	0.81	0.82	0.82	0.82	81.47
CNN-GRU	0.92	0.92	0.92	0.92	0.92	0.92	91.50
GRU	0.93	0.93	0.93	0.93	0.93	0.93	92.73
T-En	0.86	0.85	0.84	0.85	0.85	0.85	83.79
T-En-G	0.92	0.92	0.92	0.92	0.92	0.92	91.62
TransDBC	**0.96**	**0.96**	**0.95**	**0.96**	**0.95**	**0.96**	**95.38**

Table 5.3: Comparison with baseline models on UAH-DriveSet Dataset

RQ5: Hyperparameter Sensitivity of TransDBC We evaluated the hyperparameter sensitivity of the model in various settings on five different datasets as follows:

- **Effect of number of layers ($\mathbf{N_e}$, $\mathbf{N_d}$)** In our experiments we started from single encoder-decoder layer (N_e, N_d) and after every run we increased the number of layer proportionally in N_e and N_d.

Dataset ↓ / Model →	MCNN [29]	t-LeNet [68]	MCDCNN [69]	Time-CNN [70]	TWIESN [71]	MLP	GTN [64]	TransDBC
JapaneseVowels [57]	97.6	9.2	23.8	94.4	95.6	97.6	96.5	**97.71**
ArabicDigits [58]	10	10	95.9	95.8	85.3	96.9	**98.8**	98.5
KickvsPunch [59]	54.0	50	56.0	62	67.0	61	90	**100**
CharacterTraj. [60]	5.4	6.7	93.8	96	92	96.9	**97**	96.7
Wafer [59]	89.4	89.4	65.8	94	94.9	89.4	99.1	**99.3**

Table 5.4: Comparison with state-of-the-art models on Five different datasets

Up to six no of layers, the performance was improving, but as we increased after that, it started to degrade the performance as shown in Figure 5.3.

- **Effect of dropout (D)** A Simple Method for Preventing Overfitting in Neural Networks. Dropout is a training strategy in which randomly selected neurons are rejected. They are "disappeared" at random. This means that on the forward pass, their contribution to the activation of downstream neurons is removed temporally, and on the backward pass, any weight updates are not applied to the neuron. We can see in Figure 5.4 that the inclusion of dropout improves the performance of the model.

- **Effect of feed-forward layer dimension ($\mathbf{FF_{dim}}$)** The feed-forward layer is made up of weights that are learned during training and applied to each token position using the same matrix. It is a highly parallelizable portion of the model since it is applied without any communication with or inference from other token positions. The duty and aim of this attention layer is to process the output of the previous attention layer in order to better fit the input of the following attention layer. We can see in Figure 5.5 that on dimension 32 of the feed-forward layer, our model is giving the best performances.

- **Effect of parallel attention layers ($\mathbf{N_{head}}$)** The attention module of the transformer performs its calculations numerous times in parallel. An attention head is a name given to each of these. The attention module separates its Query, Key, and Value arguments N_{head} times and sends each split to a different Head. Our model performs better when the parallel attention layers are three, as shown in Figure 5.6.

5.2 A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability

To effectively illustrate the real-world topology, we use OpenStreetMap data because it is freely available, and it is easy to extract the map of the entire city. The map is visualized using the traffic simulator SUMO

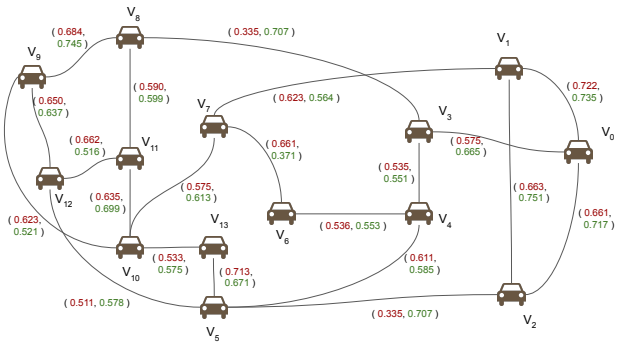


Figure 5.7: Mean Link Weight Graph

due to the availability of a graphical user interface. Various *.xml* files (i.e., city.osm.xml, city.poly.xml, city.net.xml etc.) is used to pre and post-process the raw data of the city map downloaded from the OpenStreetMap. Using the tools provided by SUMO, we post-process the obstacle data and extract useful information and the geographical position of each obstacle. A stochastic car-to-car following model is used to control the mobility of the network. Moreover, the proposed model is simulated on OMNeT++, a modular, component-based framework built for the network simulations. Python programming language is used for scripting DRL environments and linked to the OMNeT++ framework. All the experiments are conducted on an Intel i5 4.2 GHz processor with 8 Gb of memory. We assume that each vehicle is equipped with the OBU and can communicate in the range of 100-300m. The SDN layer is used to provide centralized authorization over the network. The SDN controller can communicate with the RSUs in the network through the LTE network. The RSUs are connected with the vehicle using the DSRC protocol.

5.2.1 Interpretability

The critical connections search method is implemented to analyze the interpretability of the obtained hypergraph, which computes the importance of each link between two vehicular nodes in a routing decision in the form of mean link weight or mask value. The mask value indicates the preference of a link in a routing path where the higher weight represents a shorter link or a less congested link compared to the low-weighted link. The mean link weights obtained for our case are shown in Fig. 5.7. The figure contains mask values mentioned besides an edge/link. The first value on an edge corresponds to the mask value of the link having source as lower-numbered vehicular node and destination as a higher-numbered vehicular node in red color. In contrast, the mask value for the reverse link is mentioned in green color. For example, node V_0 is a lower-numbered node than node V_1, 0.722 is the mask value of the link $V_0 - V_1$, and 0.735 is the mask value of the link $V_1 - V_0$. The mask value for the most critical links is very close to one.

5.2.2 Computational Time Analysis

For the hypergraph-based model, the computation time is very low. The total time taken to obtain hypergraph representation is 130.12 seconds on average, which is insignificant compared to the time required to learn a deep neural network that takes a few days to achieve. In our case also, the CNN-GRU model took a few days in the training phase. Further, due to the fewer parameters and less computations involved in the hypergraph-based model, the computation time required to calculate the overall optimal routing path is reduced by 31.35 milliseconds for 14 communication nodes. This reduction in time is significant in the case of resource constraint environments such as VANETs since there are many scenarios where a quick response is required, such as in case of RSUs, it may have to process the request from various vehicles at a given point of time. Table 5.5 shows the SDN processing time of various models computed for 14 communication nodes and averaged over 150 routing paths.

Model	**Processing Time (ms)**
CNN-GRU based model	63.44
Hypergraph based model	32.09
Reinforcement Routing based model [36]	55.69
CNN based model [17]	51.26
DNN based model [33]	48.93

Table 5.5: Computation Time Calculation

6 Summary and Future plan of work

6.1 TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification

This project work introduces a historical smartphone telematics multivariate time-series data-based driver behavior classification system. It is challenging for the driver to understand whether his driving is dangerous or not or what his behavior is with respect to the actions on the vehicle. Many times they become habitual to drive in hazardous mode. This system will help to avoid unsafe driving situations by the drivers. Our findings indicate that smartphone telematics data can provide a measure for driver behavior identification with 96% of average weighted precision, recall, F-measure, and 95.38% accuracy using the transformer network. We have compared our results with the baseline, state-of-the-art, and transformer-based models. The proposed framework will provide safer, more comfortable, and cheap systems making future cars capable of telematics monitoring.

We are currently considering 9 channel sensor data that captures positional and vehicle-related information such as speed, acceleration, yaw, roll, pitch, number of vehicles, and distance to the ahead vehicle to predict driver behavior. In future work, we plan to incorporate more vehicle-related sensor data, driver's medical history, and environmental factors such as weather, temperature, and humidity to provide a more accurate and robust driver behavior prediction.

6.2 A Deep Learning based Model for Spatio-Temporal Correlation in Software-Defined Vehicular Networks using Hypergraph: Towards Enhanced Interpretability

This project work proposed a scheme that exploits the structure of hypergraphs to resolve the issue of interpretability and heavyweight deployment in the optimal route identification problem based on deep learning models. The approach initially implemented a CNN-GRU model to improve network performance and finding spatio-temporal correlations. The CNN-GRU model learning is based on the DDPG technique. The packet forwarding and destination delivery parameters are used as input to the reward function. It was observed that CNN-GRU provides optimum solutions, and DRL helps to perform the route discovery task efficiently. Further, the efficient representations learned by the CNN-GRU model are used to formulate a hypergraph representation. The hypergraph representation achieved the required level of interpretation. Results show that the proposed approach is able to reduce the processing time by 33% against DNN based models. Even though the proposed solution provides high network performance with interpretable decisions, It does not consider the environmental conditions such as weather, road conditions and obstacles as the input features to the model. In the future, we plan to involve the real world traffic behavior-related features in the learning.

Publications

J. Vyas, N. Bhardwaj, Bhumika, and D. Das. "TransDBC: Transformer for Multivariate Time-Series based Driver Behavior Classification." In 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022.

References

[1] WHO, "Road traffic injuries," Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries", [Online] (Last accessed : 13-07-2021).

[2] M. of Road Transport and Highways, "Road accidents in india," Available: https://morth.nic.in/road-accident-in-india, [Online] (Accessed: 14-Jul-2021).

[3] D. Dmitrenko, E. Maggioni, G. Brianza, B. E. Holthausen, B. N. Walker, and M. Obrist, "Caroma therapy: pleasant scents promote safer driving, better mood, and improved well-being in angry drivers," in *Proceedings of the 2020 chi conference on human factors in computing systems*, 2020, pp. 1–13.

[4] R. Verma, B. Mitra, and S. Chakraborty, "Avoiding stress driving: Online trip recommendation from driving behavior prediction," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom.* IEEE, 2019, pp. 1–10.

[5] A. U. Nambi, S. Bannur, I. Mehta, H. Kalra, A. Virmani, V. N. Padmanabhan, R. Bhandari, and B. Raman, "Hams: Driver and driving monitoring using a smartphone," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 840–842.

[6] J. Vyas, D. Das, and S. K. Das, "Vehicular edge computing based driver recommendation system using federated learning," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS).* IEEE, 2020, pp. 675–683.

[7] J. Vyas, D. Das, and S. Chaudhury, "Drivebfr: Driver behavior and fuel efficiency-based recommendation system," *IEEE Transactions on Computational Social Systems*, 2021.

[8] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 401–449, 2021.

[9] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE access*, vol. 6, pp. 1662–1669, 2017.

[10] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[11] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[13] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "Ammus: A survey of transformer-based pretrained models in natural language processing," *arXiv preprint arXiv:2108.05542*, 2021.

[14] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on visual transformer," *arXiv preprint arXiv:2012.12556*, 2020.

[15] M. Zhu, J. Cao, D. Pang, Z. He, and M. Xu, "Sdn-based routing for efficient message propagation in vanet," in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2015, pp. 788–797.

[16] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in sdn," in *Proceedings of the 2019 ACM Symposium on SDN Research*, 2019, pp. 140–151.

[17] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[18] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of the 2018 conference of the ACM special interest group on data communication*, 2018, pp. 191–205.

[19] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu, "Interpreting deep learning-based networking systems," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 154–171.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[21] J. Chen, Z. Xiao, H. Xing, P. Dai, S. Luo, and M. A. Iqbal, "Stdpg: a spatio-temporal deterministic policy gradient agent for dynamic routing in sdn," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[22] A. Pjetri, M. Simoncini, F. Sambo, A. Lori, and F. Schoen, "Light footprint driving behaviour classification," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1186–1191.

[23] E. Carvalho, B. V. Ferreira, J. Ferreira, C. De Souza, H. V. Carvalho, Y. Suhara, A. S. Pentland, and G. Pessin, "Exploiting the use of recurrent neural networks for driver behavior profiling," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3016–3021.

[24] M. A. Khodairy and G. Abosamra, "Driving behavior classification based on oversampled signals of smartphone embedded sensors using an optimized stacked-lstm neural networks," *IEEE Access*, vol. 9, pp. 4957–4972, 2021.

[25] S. M. Kouchak and A. Gaffar, "Using bidirectional long short term memory with attention layer to estimate driver behavior," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 315–320.

[26] Y. Moukafih, H. Hafidi, and M. Ghogho, "Aggressive driving detection using deep learning-based time series classification," in *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*. IEEE, 2019, pp. 1–5.

[27] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "Tapnet: Multivariate time series classification with attentional prototypical network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6845–6852.

[28] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *International conference on web-age information management*. Springer, 2014, pp. 298–310.

[29] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[30] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.

[31] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 95 397–95 417, 2019.

[32] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[33] J. Reis, M. Rocha, T. K. Phan, D. Griffin, F. Le, and M. Rio, "Deep neural networks for network routing," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[34] M. Saravanan and P. Ganeshkumar, "Routing using reinforcement learning in vehicular ad hoc networks," *Computational Intelligence*, vol. 36, no. 2, pp. 682–697, 2020.

[35] L. Xin, W. Song, Z. Cao, and J. Zhang, "Step-wise deep learning models for solving routing problems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4861–4871, 2020.

[36] L. Luo, L. Sheng, H. Yu, and G. Sun, "Intersection-based v2x routing via reinforcement learning in vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[37] A. Nahar and D. Das, "Adaptive reinforcement routing in software defined vehicular networks," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2020, pp. 2118–2123.

[38] C. Yu, J. Lan, Z. Guo, and Y. Hu, "Drom: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64 533–64 539, 2018.

[39] D. Zhang, T. Zhang, and X. Liu, "Novel self-adaptive routing service algorithm for application in vanet," *Applied Intelligence*, vol. 49, no. 5, pp. 1866–1879, 2019.

[40] Y. Tang, N. Cheng, W. Wu, M. Wang, Y. Dai, and X. Shen, "Delay-minimization routing for heterogeneous vanets with machine learning based mobility prediction," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3967–3979, 2019.

[41] A. Di Maio, M. R. Palattella, and T. Engel, "Multi-flow congestion-aware routing in software-defined vehicular networks," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[42] M. Chahal and S. Harit, "Network selection and data dissemination in heterogeneous software-defined vehicular network," *Computer Networks*, vol. 161, pp. 32–44, 2019.

[43] K. S. K. Liyanage, M. Ma, and P. H. J. Chong, "Connectivity aware tribrid routing framework for a generalized software defined vehicular network," *Computer Networks*, vol. 152, pp. 167–177, 2019.

[44] W. Qi, Q. Song, X. Wang, L. Guo, and Z. Ning, "Sdn-enabled social-aware clustering in 5g-vanet systems," *IEEE Access*, vol. 6, pp. 28 213–28 224, 2018.

[45] J. Bhatia, R. Dave, H. Bhayani, S. Tanwar, and A. Nayyar, "Sdn-based real-time urban traffic analysis in vanet environment," *Computer Communications*, vol. 149, pp. 162–175, 2020.

[46] J. Wang, K. Liu, K. Xiao, X. Wang, Q. Han, and V. C. S. Lee, "Delay-constrained routing via heterogeneous vehicular communications in software defined busnet," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5957–5970, 2019.

[47] S. Din, A. Paul, and A. Rehman, "5g-enabled hierarchical architecture for software-defined intelligent transportation system," *Computer Networks*, vol. 150, pp. 81–89, 2019.

[48] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8827–8836.

[49] T. Feng and C. Yue, "Visualizing and interpreting rnn models in url-based phishing detection," in *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*, 2020, pp. 13–24.

[50] L. Arras, J. Arjona-Medina, M. Widrich, G. Montavon, M. Gillhofer, K.-R. Müller, S. Hochreiter, and W. Samek, "Explaining and interpreting lstms," in *Explainable ai: Interpreting, explaining and visualizing deep learning*. Springer, 2019, pp. 211–238.

[51] T. Linzen, G. Chrupała, and A. Alishahi, "Proceedings of the 2018 emnlp workshop blackboxnlp: Analyzing and interpreting neural networks for nlp," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.

[52] V. Mitra and H. Franco, "Interpreting dnn output layer activations: A strategy to cope with unseen data in speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5724–5728.

[53] H. D. Nguyen, X.-S. Vu, and D.-T. Le, "Modular graph transformer networks for multi-label image classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9092–9100.

[54] J. Zhang, W.-c. Chang, H.-f. Yu, and I. Dhillon, "Fast multi-resolution transformer fine-tuning for extreme multi-label text classification," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[55] P. Li, P. Zhong, K. Mao, D. Wang, X. Yang, Y. Liu, J. Yin, and S. See, "Act: an attentive convolutional transformer for efficient text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 261–13 269.

[56] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? presenting the public uah-driveset," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 387–392.

[57] JapaneseVowel, "Japanese vowels data set," Available: https://archive.ics.uci.edu/ml/datasets/Japanese+Vowels, [Online].

[58] ArabicDigits, "Spoken arabic digit data set," Available: https://archive.ics.uci.edu/ml/datasets/Spoken+Arabic+Digit, [Online].

[59] M. G. Baydogan, "Multivariate time series classification datasets," 2019.

[60] CharacterTrajectories, "Character trajectories data set," Available: https://tinyurl.com/bdfjxp5x, [Online].

[61] K. Saleh, M. Hossny, and S. Nahavandi, "Driving behavior classification based on sensor data fusion using lstm recurrent neural networks," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[62] J. Xie, A. R. Hilal, and D. Kulić, "Driving maneuver classification: A comparison of feature extraction methods," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 4777–4784, 2017.

[63] D. Yi, J. Su, C. Liu, M. Quddus, and W.-H. Chen, "A machine learning based personalized system for driving state recognition," *Transportation Research Part C: Emerging Technologies*, vol. 105, pp. 241–261, 2019.

[64] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, "Gated transformer networks for multivariate time series classification," *arXiv preprint arXiv:2103.14438*, 2021.

[65] K. Schlegel, F. Mirus, P. Neubert, and P. Protzel, "Multivariate time series analysis for driving style classification using neural networks and hyperdimensional computing," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 602–609.

[66] A. El Mekki, A. Bouhoute, and I. Berrada, "Improving driver identification for the next-generation of in-vehicle software systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7406–7415, 2019.

[67] J. Xie and M. Zhu, "Maneuver-based driving behavior classification based on random forest," *IEEE Sensors Letters*, vol. 3, no. 11, pp. 1–4, 2019.

[68] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *ECML/PKDD workshop on advanced analytics and learning on temporal data*, 2016.

[69] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 96–112, 2016.

[70] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[71] P. Tanisaro and G. Heidemann, "Time series classification using time warping invariant echo state networks," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 831–836.