AutoWatch: Learning Driver Behavior with Graphs for Auto Theft Detection and Situational Awareness

Paul Agbaje*, Abraham Mookhoek*, Afia Anjum*, Arkajyoti Mitra*, Mert D. Pesé†, Habeeb Olufowobi*
*Department of Computer Science and Engineering, University of Texas at Arlington
†School of Computing, Clemson University
{pauloluwatowoju.agbaje, habeeb.olufowobi}@uta.edu

***Abstract*—Millions of lives are lost due to road accidents each year, emphasizing the importance of improving driver safety measures. In addition, physical vehicle security is a persistent challenge exacerbated by the growing interconnectivity of vehicles, allowing adversaries to engage in vehicle theft and compromising driver privacy. The integration of advanced sensors with internet connectivity has ushered in the era of intelligent transportation systems (ITS), enabling vehicles to generate abundant data that facilitates diverse vehicular applications. These data can also provide insights into driver behavior, enabling effective driver monitoring to support safety and security. In this paper, we propose AutoWatch, a graph-based approach for modeling the behavior of drivers, verifying the identity of the driver, and detecting unsafe driving maneuvers. Our evaluation shows that AutoWatch can improve driver identification accuracy by up to 22% and driving maneuver classification by up to 5.7% compared to baseline techniques.**

***Index Terms*—Driver Behavior, Situational Awareness, Graphs, Graph Neural Network**

I. Introduction

Increasingly, the risk of vehicle theft continues to be a concern for car owners. The National Highway Traffic Safety Administration (NHTSA) reports that over 1 million vehicles were stolen in the United States (U.S.) in 2022 alone, with over 74% being passenger vehicles [1]. Thus, every 32 seconds, a vehicle is stolen in the U.S. This statistic is even more likely to be lower than the actual state of things as the Federal Motor Vehicle Theft Prevention Standard by the NHTSA also admitted that not all thefts are reported, making the overall estimation of motor vehicle thefts difficult [2].

Modern vehicles, with their embedded sensors and Internet connectivity, have evolved into platforms that facilitate various intelligent transportation system (ITS) applications, such as real-time traffic navigation, autonomous driving, advanced driver assistance systems, and parking management. Nevertheless, the growing interconnectedness among vehicles presents opportunities for malicious actors to exploit vehicular vulnerabilities, opening up new car theft opportunities. For instance, car thieves can leverage weaknesses in a vehicle's remote key entry system to gain access, bypass the immobilizer system, or program a new key through the onboard diagnostics (OBD)-II port [3]. Through an investigation into actual car theft in 2022, Ken Tindell [4] showed that keyless car theft is possible with a controller area network (CAN) injector grafted onto a James Bullough Lansing (JBL) speaker's circuit board. The attackers accessed the CAN bus from the headlight connector and injected successful frames onto the bus before stealing the car. Beyond the loss of vehicles, the rise of data collection and processing platforms based on vehicular telematics data raises significant privacy concerns since the data must not be shared with malicious entities [5]. Car theft can lead to data breaches and potential misuse of personal information [6].

Apart from the challenge of auto theft, the World Health Organization (WHO) reported that approximately 1.3 million lives are lost annually, and an additional 20-50 million people suffer injuries due to road accidents. Many individuals endure long-term disabilities due to these accidents [7]. Fig. 1 illustrates the mortality count resulting from vehicle crashes spanning 1975 to 2021, based on data from the U.S. Department of Transportation [8], [9]. We observe that the mortality rate initially declined as the development of transportation technology kept improving. However, the proliferation of sensors and interconnectedness in recent times across the vehicular industry also exposed several attack surfaces for an adversary. During the year 2019, the U.S. recorded a total of 36,355 fatalities due to vehicular accidents. Despite the pandemic, this number increased to 39,007 in 2020 and 42,939 in 2021. Two possible reasons for the increase in fatality are the increased propensity for risky driving behaviors and reduced law enforcement on roads during this period [10]. The fatality rate underscores the importance of comprehensive initiatives that can improve road safety. An example of such an initiative is Vision Zero [11], dedicated to enhancing road safety with the ambitious vision of eliminating road-related fatalities and severe injuries. With initiatives like Vision Zero, there is potential to improve overall road safety significantly and enhance the effectiveness of ITS.

The advancements in vehicular connectivity have allowed vehicles to produce extensive data that support safety and security applications in ITS [12]. These data are sourced from various components, including the in-vehicle CAN [13], [14] and a range of sensors, such as gyroscopes and accelerometers, even smartphones, which can be leveraged to learn and model

Symposium on Vehicles Security and Privacy (VehicleSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-7-6
https://dx.doi.org/10.14722/vehiclesec.2024.23072
www.ndss-symposium.org

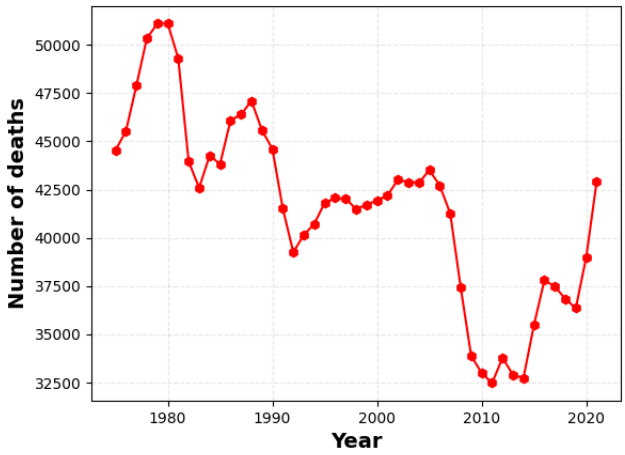

Fig. 1. Total number of deaths in vehicle crashes from 1975-2021

driver behavior [15]. This behavior model is then helpful for monitoring drivers to facilitate safety and mitigate vehicle theft. For instance, collecting data representing a driver's unique driving characteristics, such as driving speed, steering angle, and sudden acceleration rates, can support driver profiling applications to verify the identity of the car owner [16]. In addition, these behavior models can be used for situational awareness, allowing the detection of risky and improper driving maneuvers that jeopardize the safety of road users [17]. Efficiently recognizing these risky driving maneuvers can be crucial to safe driving in congested driving conditions, where sudden and unexpected driving behaviors can lead to collisions with other vehicles on the road.

To improve road safety, physical vehicle security, and driver privacy, we propose a graph-based technique, AutoWatch, that efficiently models the relationships between the driving behaviors of different drivers to identify car owners and detect improper driving behavior. Concretely, AutoWatch extracts feature components with the most impact for efficient clustering to establish links by learning the relationship between different driving behaviors. Using this learned behavior, we predict the car owner for identification verification and classify the driver's behavior to support situational awareness.

In summary, we make the following contributions in this paper:

- We propose AutoWatch, a graph-based approach that models driving characteristics as a graph, where each node represents a driver or a unique driving behavior.
- To learn the relationship among different driving characteristics, we use a lightweight unsupervised learning approach based on principal component analysis (PCA), KMeans clustering, and horizontal visibility graph (HVG). Using these approaches, we capture the dependencies and associations among the diverse data features.
- We implement a graph neural network (GNN)-based approach for driver identification and driving maneuver classification to support safety and security in ITS.
- To demonstrate the effectiveness of our approach, we evaluate the performance using publicly available vehicular datasets and compare our work with baseline models. Our results show that AutoWatch can support vehicular safety and security in ITS.

II. Background

This section provides an overview of our approaches for establishing connections among data points, including dimensionality reduction, clustering, and visibility graph-based techniques. Additionally, we introduce GNNs, which we use to learn the embeddings that encode the relationships among different data points.

A. Dimensionality Reduction

High dimensional data can introduce the curse of dimensionality, where data points become equidistant from each other, adversely affecting distance-based clustering techniques [18]. Dimensionality reduction techniques help transform these data points from a high-dimensional to a lower-dimensional space. The objective is to retain the essential properties present in the original data while reducing the number of features. The technique helps derive new representative features from an input dataset. These new features can be beneficial in clustering different data points to improve accuracy results and address the challenges posed by high-dimensional data, which slow down the convergence of clustering algorithms [19], [20]. An example of a dimensionality reduction technique is Principal Component Analysis (PCA), a statistical method used to convert the original variables from an input dataset to a new set of linear combinations that maximally preserves the covariance of the original data [21].

B. Clustering

Clustering is an unsupervised learning process for finding meaningful groupings within a set of unlabelled data. The technique leverages the inherent structure in the data to isolate groups with common traits or attributes and identify patterns and relationships between data points [22]. This partitioning enables further exploration of the relationships between these groups and aids in uncovering insights from the data. A well-known clustering method is the KMeans clustering algorithm [23], which partitions n data points into k clusters, where each cluster represents a group of data points with high similarity. The KMeans algorithm iteratively assigns data points to the nearest cluster based on a distance metric. This technique can help group data points to extract insightful information.

C. Visibility Graphs

A visibility graph uses graph theoretical concepts to transform time series data into a graph representation useful for analyzing interrelations among associated data points in the graph [24]. In the generated graph, data points are represented as nodes. Two arbitrary data in the graph, such as (t_k, y_k) and (t_n, y_n) are connected, if any other data (t_m, y_m) between them fulfils the following:

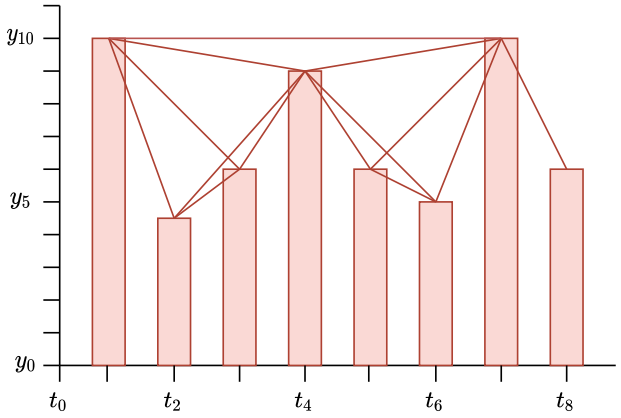

Fig. 2. An example of time series data with the associated connection using visibility graphs.

$$y_m < y_n + (y_k - y_n)\frac{t_n - t_m}{t_n - t_k} \tag{1}$$

The associated graph generated from the time series data is always connected, undirected, and invariant under affine transformations of the series data [25]. Fig. 2 shows an application of the visibility graph technique. The bars in the graph represent the data values y_a where $y_0 \leq y_a \leq y_{10}$ occurring at time t_a where $t_0 \leq t_a \leq t_{10}$. The visibility rays in the graph represent the connecting links between different data points in the time series. If the first three data points in the graph are represented as $(t_k, y_k), (t_m, y_m), (t_n, y_n)$ with values $(1, 10), (2, 4.5), (3, 6)$, respectively, following Equation 1, $y_m = 4.5 < 6 + (10 - 6) \times \frac{3-2}{3-1} = 8$, which satisfies the criterion. Therefore, the first and third data points, (t_k, y_k) and (t_n, y_n), are connected. Data points are generally connected in the resulting graph if the visibility rays do not intercept another data point placed between them.

D. Graph Neural Networks

A graph is a data structure representing entities as nodes and their relationships as edges [26]. Using graph representations, we represent a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V}$ signifies the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ characterizes the observed links, and $\mathcal{X}$ corresponds to the attributes matrix. In addition, the graph's adjacency matrix is symbolized as $\mathcal{A}$, where $\mathcal{A}_{uv} \in \mathcal{A}$, with u and v denoting nodes. If there is an edge connecting nodes u and v, $\mathcal{A}_{uv} = 1$, otherwise, $\mathcal{A}_{uv} = 0$. Furthermore, $\mathcal{X}_u \in \mathcal{X}$, $\mathcal{X}_u$ representing the attributes of node u, and $X \in \mathbb{R}^{|V| \times f}$, where f represents the feature dimension for each node. Graph Neural Networks (GNNs) generate node embeddings based on an attributed graph's attributes and topological structure. For a given graph $\mathcal{G}$, with its structural and feature information as input, a GNN model denoted as $\mathcal{F}(\cdot)$ produces node embeddings in a high-dimensional space, resulting in $\mathcal{Z} \in \mathbb{R}^{|V| \times h}$, where h represents the dimensionality of the node embeddings. These learned embeddings $\mathcal{Z}$ serve as a foundation for various tasks, including node prediction.

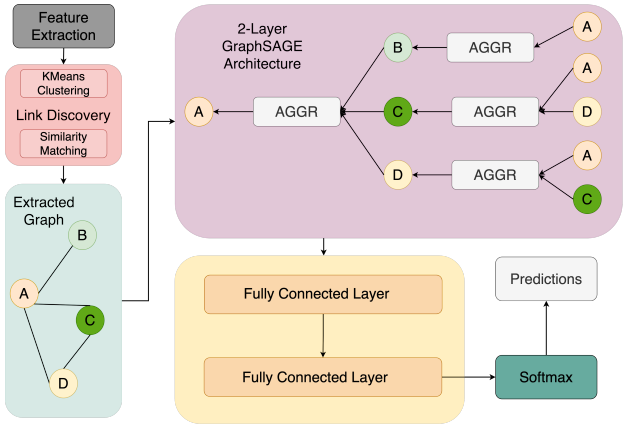

Fig. 3. Overview of AutoWatch with a GraphSAGE architecture of depth 2 and a 2-layer multi-layer perceptron (MLP) architecture. AGGR represents the AGGREGATE function.

III. System Design

This section outlines the design overview of our proposed approach, AutoWatch. First, we explore how AutoWatch completes its feature selection technique for reducing the input dataset's dimensionality. Subsequently, we discuss the methodology used by AutoWatch to establish links among different data points and explain the graph-based approach used for the downstream prediction tasks—the classification tasks that depend on the link discovery capabilities of AutoWatch. Fig. 3 shows the architecture of AutoWatch. Following feature extraction using PCA, link discovery—a technique we use to find relevant relationships among different data points—is performed using KMeans clustering and cosine-based similarity matching. The extracted graph from the link discovery process is fed as input to a GraphSAGE architecture with a depth of 2. The output from GraphSAGE passes to a downstream multi-layer perceptron (MLP) classifier. Finally, the MLP output logits pass through the softmax function to give the probability distribution of the predicted classes.

A. Feature Extraction

Vehicle data can be high-dimensional, with many attributes offering different information, such as speed, acceleration, and steering wheel angle metrics. Understanding the salient relationship between these data points is necessary to identify the individual behind the wheel and classify the driver's maneuvers on the road. However, high-dimensional data can affect clustering performance, essential for establishing relationships among data points. Consequently, understanding the individual importance of each feature is crucial to AutoWatch's design, as it aims to reduce the dimension of the data for efficient clustering. To determine the most critical driver behavior features, we used PCA to transform the original high-dimensional data points into new variables in a smaller subspace. These new variables, called principal components (PCs), summarize the data without losing vital information present in the original data. Also, PCs are linear combinations of the original data features that encapsulate the maximum variance in the data. Although fewer components can be

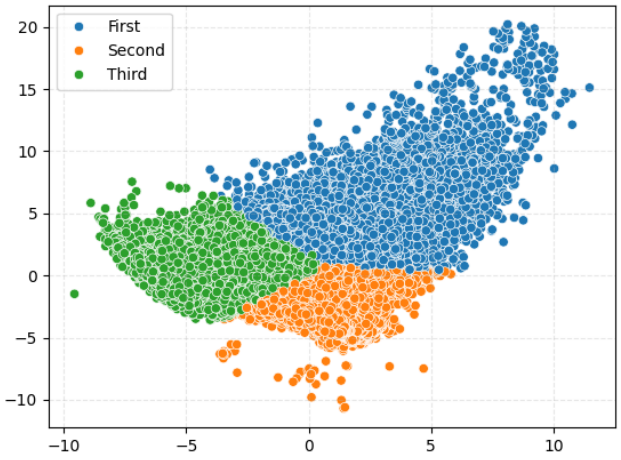

Fig. 4. Clustering visualization using the first two PCs. First, Second, and Third represent three different data clusters.

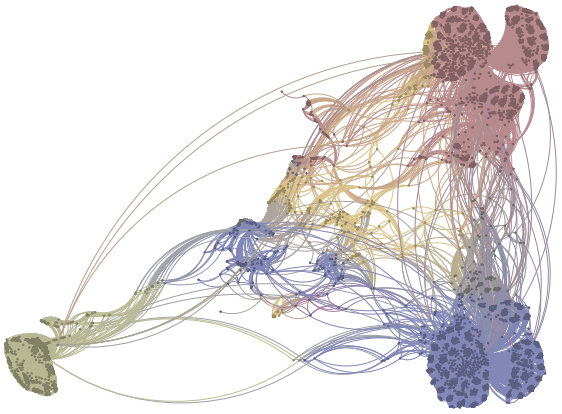

Fig. 5. Visualization of connected nodes associated with the driver identification dataset.

beneficial, their significance diminishes if these components do not capture enough information about the original data. Thus, choosing the right amount of PCs is crucial to having enough information for clustering. The explained variance ratio (EVR) [27] helps to understand the contribution of each PC by measuring the fraction of variance explained by that component [20]. We consider the minimum number of components whose cumulative EVR exceeds a predefined threshold τ. Based on our experiments, we set $\tau = 0.7$ with 10 components for clustering, choosing minimum values that do not degrade the performance of the model.

B. Link Discovery

To establish connections among different data points, we first separate the data into multiple clusters using KMeans clustering, resulting in multiple partitions of the original data based on their similarities. To simplify our design, we divide our data points into G clusters and initialize the KMeans algorithm with greedy KMeans++, which allows the clustering algorithm to use several trials at each sampling step to select the best initial cluster centroids to speed up convergence. We choose minimum values of G for both classification tasks that efficiently divide the data points based on similarity. Based on our experiments, we choose $G = 3$ and $G = 5$ for driver identification and driving maneuver classification, respectively. A visualization of the 3 clusters used for driver identification is shown in Fig. 4.

With the data successfully clustered, we assign a unique ID to each data point to track the connections among different data points. We establish connections using two techniques: cosine similarity matching and horizontal visibility graphs.

1) Cosine Similarity Matching: The cosine similarity metric is a common similarity measure in machine learning applications [28], [29]. The cosine similarity scores between two vectors α and β can be expressed as $S_s = \frac{\langle \alpha, \beta \rangle}{\|\alpha\| \|\beta\|}$. Where $\langle \alpha, \beta \rangle$ gives the dot product between vectors α and β. $\|\alpha\| \|\beta\|$ is the product of the norms of the vectors. We assume that during the design phase of AutoWatch, an existing budget B determines the number of sampled nodes to be used for establishing connections, where $0 \leq B \leq N$. Here, N is the total number of data points in the dataset, when $B = 0$, no connections are being established using AutoWatch, and when $B = N$, all available nodes are used to establish connections. To establish connections, we sample $\frac{B}{C}$ number of data points from each available class in the dataset, where C is the number of available classes. After sampling, we look for inter-cluster similarities and intra-cluster relationships between selected data points. Inter-cluster similarity helps us establish connections between data points of different clusters, while intra-cluster similarity helps us achieve similarity between data points of the same cluster. To establish a connection, we compare each node (i.e., each data point) with other sampled nodes and establish connections between pairs with the highest S_s.

2) Horizontal Visibility Graph (HVG): We also establish connections among data points within each class using the HVG algorithm, which maps the time series data of each class into graphs with established connections among nodes. The HVG algorithm simplifies the original visibility graph algorithm, and its output is always a subgraph of the more complex visibility graph [30]. Concretely, two data points are connected if we can draw a horizontal ray connecting both data points without intersecting any intermediate data point. If we represent the time series data within a class as $x_{t\,1,\ldots,N}$ and consider two points in the time series, t and $i = t + z$, where z is an integer greater than 0, then the connection between points t and i is subject to the following criteria:

$$x_t, x_i > x_n, \forall n \in N | t < n < i \tag{2}$$

Using the cosine similarity matching and the HVG techniques, we establish connections among different data points representing driver behaviors, which we use in the graph neural network that learns the embedding that can be used for classification from the supplied graphs. A visualization of the connected nodes after link discovery for the driver identification dataset we used for evaluation is shown in Fig. 5.

C. Graph-based Learning

We adopt GraphSAGE [31] to learn the relationships between connected data points and generate embeddings that are fed as input to a classifier for driver identification and driving maneuver predictions. Using an inductive learning technique, GraphSAGE leverages node features to learn embeddings that generalize to nodes that are not available during training. Other advantages of GraphSAGE include its efficiency for large graphs and its ability to learn spatial information from the underlying data by considering the topological structure of the neighbors of each node and the distribution of the features of nodes in the neighborhood [32]. To generate embeddings, GraphSAGE learns the parameters of K aggregator functions, expressed as $\text{AGGREGATE}_k, \forall k \in \{1, 2, \ldots, K\}$ and K weight matrices, expressed as $W^k, \forall k \in \{1, 2, \ldots, K\}$. These functions are used for aggregating information from the neighbors of each node $v \in \mathcal{V}$, and the weights are used to propagate information between layers of the model. Each node v generates an initial input node feature $h_v^0 \leftarrow x_v$ and for each aggregator generates an embedding for each node in its immediate neighborhood $\mathcal{N}(v)$, as follows:

$$h_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \quad (3)$$

where h_u^{k-1} is the current representation of the aggregate neighbors' embeddings, which depends on the previous iteration. The aggregated representation is concatenated with the node's current representation h_v^{k-1} as follows:

$$h_v^k \leftarrow \sigma(W^k \cdot \text{CONCAT}(h_v^{k-1}, h_{\mathcal{N}(v)}^k)) \quad (4)$$

where σ is a nonlinear activation function. In our work, we use the mean aggregator in line with the original GraphSAGE paper [31], which is a linear approximation of a localized spectral convolution [32], [33] and can be expressed as:

$$h_v^k \leftarrow \sigma(W^k \cdot \text{MEAN}(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in \mathcal{N}(v)\})) \quad (5)$$

The final representation z_v of a node can be expressed as:

$$z_v \equiv h_v^K, \forall v \in \mathcal{V} \quad (6)$$

GraphSAGE works by encouraging nodes close to each other to have similar representations and forcing nodes dissimilar to have different embeddings. The model achieves this objective by minimizing the following loss function:

$$J_G(z_u) = -log(\sigma(z_u^T z_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} log(\sigma(z_u^T z_{v_n})) \quad (7)$$

Here, $u, v \in \mathcal{V}$ and v co-occur near u on a fixed-length random walk—a graph traversal technique that begins at a node and moves to a neighbor with a certain probability. P_n is a negative sampling distribution of nodes in $\mathcal{V}$, Q represents the number of negative samples, and σ is the sigmoid activation function.

After learning the embeddings for each node, we feed these embeddings to an MLP, which we use as our classifier. For an input z_i, we define the encoding of the classifier as:

$$X_v = \sigma(W_m z_i + b_m) \quad (8)$$

where W_m and b_m are learnable parameters, and σ is a nonlinear activation function. We apply the softmax function to the final output of the MLP to convert the logits from the model's output into a probability distribution over the target output classes. The final prediction can be expressed as:

$$y_v = \arg\max(\text{SOFTMAX}(X_v)) \quad (9)$$

The prediction y_v, represents the respective identified classes for the driver identification and driving maneuver tasks.

IV. Experimental Setup

A. Datasets

To our understanding, no comprehensive dataset in the literature encompasses labels for driver identification and driving maneuver tasks in a unified manner. We leverage two diverse datasets aligned with each task's requirements and incorporate embeddings learned from these datasets during training to address the unique challenges of each task. Integrating the learned embeddings improves AutoWatch's overall performance, allowing it to excel in both tasks with improved proficiency.

1) Dataset 1: For driver identification, we use the driving dataset[1] from the Hacking and Countermeasure Research Lab [16]. The data was extracted from the OBD-II port of a model of KIA Motors Corporation in South Korea and includes a driving time of about 23 hours with a driving path totaling 23 km. The dataset includes 10 drivers labeled "A" to "J" with 94,401 records and 51 distinct features retrieved from recordings every 1 second while driving.

2) Dataset 2: To demonstrate the proposed approach's efficiency in extracting useful connections, which improves classification performance, we use another publicly available driving behavior dataset[2] used for modeling risky driver behaviors based on accelerometer and gyroscope data [34]. The data was extracted from experiments conducted on a Ford Fiesta 1.4, a Ford Fiesta 1.25, and a Hyundai i20 using 3 drivers aged 27, 28, and 37. Equipment used to extract the data includes an MPU6050 (3-axis accelerometer and 3-axis gyroscope) and a Raspberry Pi 3 Model B. The dataset contains 4 driving behaviors: sudden acceleration, sudden right turn, sudden left turn, and sudden break. For our experiment, we sample data with a window size of 14 seconds.

B. Data Preprocessing

We preprocess the datasets by converting the labels to numerical values and then normalizing the attributes using the z-score. The z-score scales the data to unit variance and a mean of zero, allowing the data to assume Gaussian distribution. The z-score for each data point can be expressed as $z = \frac{x-\mu}{\sigma}$.

[1]Dataset 1 is available at http://ocslab.hksecurity.net/Datasets/driving-dataset

[2]Dataset 2 is available at https://data.mendeley.com/datasets/jj3tw8kj6h/2

Where u is the mean and σ is the standard deviation of the data. x is the data point we want to standardize, and z is the standardized value.

C. Baselines

We compare the performance of our graph-based AutoWatch against four baselines: support vector machine (SVM), decision tree (DT), K-nearest neighbors (KNN), and multi-layer perception (MLP). These are the baselines used in the literature for driver identification and driving maneuver classification [16], [35]–[37].

1) Support Vector Machines (SVMs): SVMs [38] are a class of supervised learning algorithms used for classification [39]. Classifiers in SVMs use hyperplanes corresponding to the classification task's decision functions. The classifier can also perform nonlinear classification tasks made possible using kernel functions.

2) Decision Trees (DTs): DTs [40] are types of classifiers that recursively partition the input space to form a directed tree with a root node with no incoming edges and other nodes with only one incoming edge. The internal nodes of the tree represent tests on specific data attributes and the outgoing edges of these nodes represent the outcome of the respective tests [41]. The leaf nodes of the tree denote predicted class labels based on the path followed during the recursive partition.

3) K-nearest neighbors (KNN): KNN [42] is a supervised non-parametric classification technique known for its simplicity and effectiveness [43]. For classification tasks, KNN algorithms identify data that are closest in a given region using distance metrics such as Euclidean, Hamming, and Chebychev distances [44]. We have used the Euclidean distance for our comparisons since it is the most common one used in KNN [42].

4) Multi-layer perceptrons (MLPs): MLPs [45] are types of artificial neural networks that are widely used for different tasks, including classification problems. Feature vectors extracted from the training data are fed as input to the MLP, which has a network of nodes that compute simple functions [46]. The outputs of the MLP are logits that can be converted to probability distributions for classification. In our work, we specifically choose the MLP as our downstream classifier to show improvement in the performance of the classifier when used with the learned embeddings of AutoWatch.

D. Metrics

We evaluate the effectiveness of AutoWatch using standard metrics such as accuracy, precision, recall, and F1 score.

1) Accuracy: The accuracy, A, is the number of correctly predicted classes divided by the total number of observations. The accuracy can be expressed as:

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

where TN and TP are the true negative and true positive values, respectively. FN and FP are the false negative and false positive values, respectively.

TABLE I
OVERVIEW OF DATASETS AFTER LINK DISCOVERY

Datasets	No. of nodes	No. of edges	No. of features
Dataset 1	94380	58616	52
Dataset 2	1114	533	12

2) Precision: The precision, P, denotes the proportion of identified classes that were actually correct. We can express P as follows:

$$P = \frac{TP}{TP + FP} \tag{11}$$

3) Recall: The recall, R, is the fraction of correctly predicted instances of a specific class. The recall can be expressed as:

$$R = \frac{TP}{TP + FN} \tag{12}$$

4) F1 score: The F1 score offers a compact assessment of a model's effectiveness using a weighted average of the precision and the recall. The F1 score is given by:

$$F1 = \frac{2 * (R * P)}{R + P} \tag{13}$$

V. EVALUATION

In this section, we present the results of our experiment. With 10% of nodes from each dataset used for establishing connections, AutoWatch's link discovery stage successfully identified 58,616 and 533 relationships with Datasets 1 and 2, respectively. Table I summarizes the dataset configuration after the link discovery stage.

A. Performance on identification tasks

We evaluate the performance of AutoWatch on the driver identification and driving maneuver classification tasks. In our experiment, we use the same configurations for the baseline MLP model and our downstream MLP classifier for fair comparison. For Dataset 1, we use a 2-layer MLP with the rectified linear unit activation function and a learning rate of 1e-03. We also use a hidden layer with 128 hidden units. For Dataset 2, we use a hidden unit of 512 and run the models for 1,000 epochs.

The confusion matrix for the driver identification task on Dataset 1 is shown in Fig. 6. The results show that AutoWatch can efficiently identify most of the drivers based on their driving styles. Drivers "A" and "E" are recognized with an accuracy of 100%. The lowest classification results for the driver identification tasks were on drivers "D" and "G", with accuracies of 99.88% and 99.87%, respectively. Notably, a significant portion of false positives for driver "G" results from its misclassification as "D," with 0.08% misclassification accuracy on "D".

Fig. 7 shows the result for the driving maneuver task on Dataset 2. AutoWatch was able to attain 100% accuracy with

TABLE II
COMPARISON OF AUTOWATCH'S PERFORMANCE WITH 4 BASELINES AND 2 DATASETS. RESULTS ARE GIVEN IN PERCENTAGES (%)

Models/Datasets	Metrics	SVM	DT	KNN	MLP	AutoWatch
Dataset 1	**Accuracy**	85.9	81.9	90.4	99.5	**99.9**
	Precision	86.3	82.9	90.6	99.5	**99.9**
	Recall	85.9	81.9	90.4	99.5	**99.9**
	F1 Score	85.9	81.7	90.4	99.5	**99.9**
Dataset 2	**Accuracy**	94.4	97.1	94.7	98.2	**99.8**
	Precision	94.5	97.9	94.8	98.3	**99.8**
	Recall	94.4	97.9	94.7	98.2	**99.8**
	F1 Score	94.6	97.9	94.7	98.2	**99.8**

the prediction of sudden break. In addition, the model achieves also an accuracy of 100% with both sudden acceleration and sudden right turn and an accuracy of 99.32% for sudden left turn.

B. Comparison with baselines

We compare the performance of AutoWatch with other baselines used in prior work and summarize the result of our evaluation in Table II.

From the results depicted in Table II, DT has the lowest performance result for the driver identification task on Dataset 1 with an accuracy score of 81.9%. SVM improves on all four metrics, but can only achieve an accuracy of 85.9% on the identification task. The MLP performed better than the rest of the baselines achieving an accuracy of 99.5%, an improvement on the KNN which achieves an accuracy of 90.4%. AutoWatch achieves 99.9% for accuracy, precision, recall, and F1 score improving on the performance of the MLP model with only 10% of connection nodes. Overall, AutoWatch improves accuracy performance by 16.3%, 22%, 10.5%, and 0.4% for SVM, DT, KNN, and MLP, respectively. The performance of AutoWatch shows that it can better leverage the relationships between data points for different classification tasks.

Table II also shows the performance comparison for the driving maneuver task on Dataset 2. SVM achieves 94.4%, 94.5%, 94.4%, and 94.6% accuracy, precision, recall, and F1 score, respectively. KNN has a similar performance to SVM with a slight improvement, achieving an accuracy score of 94.7%, and improved by DT with an accuracy of 97.1%. AutoWatch achieves the best overall score of 99.8% for accuracy, precision, recall, and F1 score, improving the accuracy performance by 5.7%, 2.8%, 5.4%, and 1.6% for SVM, DT, KNN, and MLP, respectively. The efficiency of AutoWatch with only 10% connected nodes is more evident with dataset 2 between the base MLP classifier without graph capabilities and the AutoWatch using graph-based techniques to learn feature representations before feeding these representations to the MLP classifier.

C. Impact of number of connections

To show the impact of the number of connections on model performance, we evaluate AutoWatch on Dataset 1 using 2%, 5%, 7%, and 10% nodes for connections and running the model for 100 epochs. Fig. 8 shows the result of our evaluation. With 2% connected nodes, the model achieves an accuracy of 94.30%, which was increased to 97.96% when we used 5% nodes to establish connections. With 7% connected nodes, the accuracy improved to 98.61% and further to 99.06% with 10% connections. The result shows the importance of the number of connections for establishing crucial relationships among data points and learning embeddings that can be used for efficient classification tasks.

D. Does AutoWatch capture meaningful relationships that help with classification?

We also show the effect of AutoWatch on the resulting embeddings from the data points. For our visualization, we used the t-distributed stochastic neighbor embedding (t-SNE) [47] — a technique used for the visualization of high-dimensional data in a low-dimensional space — to reduce the dimensions of the embeddings just before the input layer of the MLP to two dimensions. Fig. 9 shows the plot of the embeddings of dataset 2 after we apply t-SNE.

It shows that AutoWatch can separate most of the data points into distinct classes, which makes it possible for a downstream classifier to classify with sufficient training data. The result shows that AutoWatch learns crucial relationships among data points that can be used to assist the downstream classifier in properly distinguishing between the different classes in the dataset.

VI. DISCUSSION

Our work demonstrates the effectiveness of AutoWatch in driver identification, contributing to reducing auto theft and enhancing situation awareness by detecting unsafe driving maneuvers. Our results show that AutoWatch facilitates generalization across various classification tasks to improve vehicle and driver safety.

The driver identification approach can be implemented by notifying the vehicle owner through a mobile application in the event of theft. Car insurance companies typically permit owners to maintain a list of authorized users [48]. Owners can establish a trust list with a trusted authority authorized to identify and flag anomalous users. Furthermore, AutoWatch enables the analysis of decision-making processes leading to different classification outcomes, revealing connections that infer similarities in the behaviors of diverse drivers.

For situational awareness, AutoWatch's application can include a dashboard that alerts the driver to anomalous driving behaviors. Similar to techniques used by usage-based insurance companies to calculate driving scores to adjust insurance premiums and discounts, the dashboard may also offer analytical insights into the driver's behavior over time [49]. This approach bolsters driver safety by leveraging available sensor information and employing our graph-based method for driving maneuver classification.

Our results show that increasing the number of connections may improve the model's efficiency. However, network designers can also consider budget constraints to determine

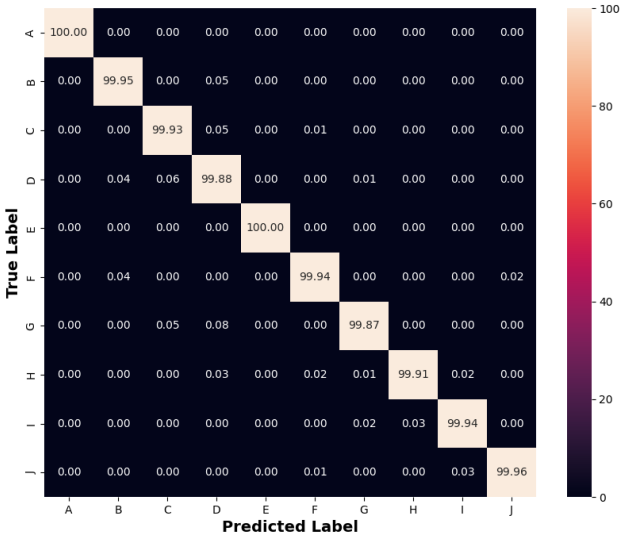

Fig. 6. Confusion matrix for Dataset 1

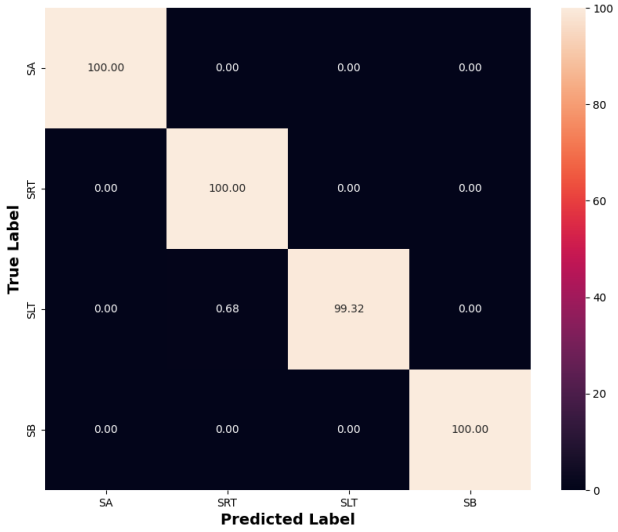

Fig. 7. Confusion matrix for Dataset 2. SB, SLT, SRT, and SA represent sudden break, sudden left turn, sudden right turn, and sudden acceleration, respectively.

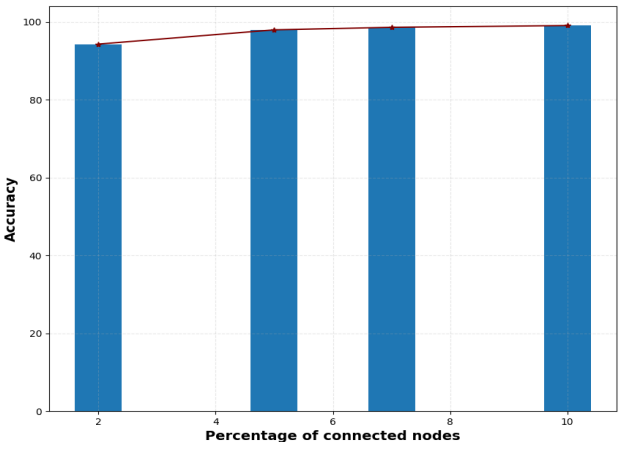

Fig. 8. Accuracy with increasing number of connections

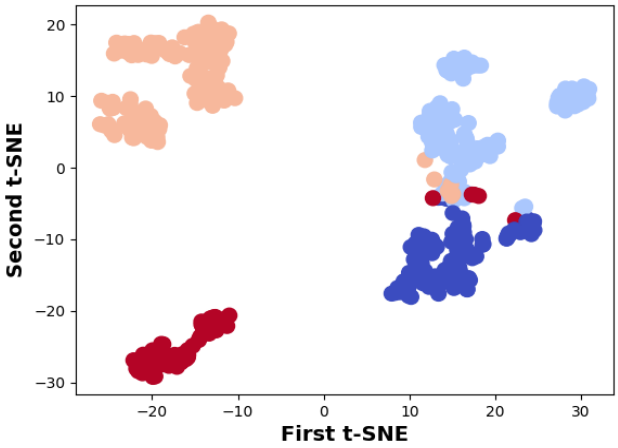

Fig. 9. A t-SNE plot of the embeddings of dataset 2. Each color represents different classes.

a trade-off between computation complexity and accuracy in choosing an optimal number of connected nodes required for classification tasks. To use the optimal number of connections, designers can decide to train AutoWatch centrally in the cloud. However, if on-device training is preferred, where the training is completed in the vehicle, the number of connections can be reduced to account for the resource constraints of individual vehicles.

Using AutoWatch, instances where time constraints are critical can leverage techniques like knowledge distillation (KD) [50], [51] to allow training a simpler student model, such as a basic MLP, for real-time classification. This method enhances the performance of resource-efficient student models by leveraging more powerful teacher models. The student model operates independently during inference, ensuring effective classification for time-sensitive tasks. In vehicles, a simpler MLP trained with a teacher AutoWatch model can be used to flag anomalous drivers and unsafe driving behaviors.

VII. Related Work

Several works have been proposed for detecting unsafe driving maneuvers by learning driver behavior through vehicular sensor data. Lattanzi *et al.* [17] proposed a methodology based on machine learning techniques, such as SVM and artificial neural network (ANN), aimed at recognizing safe and unsafe driving behaviors by taking advantage of sensors present in modern cars. Khosravi *et al.* [52] utilized real-time smartphone sensor signals by developing an Android application that collects these smartphone sensor signals to detect driving events. The collected data undergo a novel multi-classifier fusion framework, ensembling hybrid convolutional neural network (CNN), SVM, and MLP models, where each classifier processes each sample input and combines the outputs using a majority vote. Khosravinia *et al.* [53] proposed a framework

that extracts relevant sensor information from the CAN bus and uses graph convolutional recurrent neural networks deployed at the edge server to classify driving behaviors into safe and unsafe classes. In addition, the authors developed a dashboard enabling drivers to access their driving reports, monitor the prediction results, and get alerts during unsafe driving scenarios. Ma *et al.* [54] proposed a real-time abnormal driving behavior detection method based on vehicle kinematic data called the long short-term memory residual (LSTM-R). The LSTM-R method compares current and historical vehicle kinematic data, such as speed, acceleration, direction, and position, to detect unusual behavior. LSTM-R flags abnormal driving by looking at these differences over a specific time frame. While the discussed papers [17], [52]–[54] distinguish between safe and unsafe driving patterns, they overlook driver identification, a crucial factor in preventing vehicle theft. Our proposed method efficiently models the relationships between the driving behaviors of different drivers to identify car owners along with detecting unsafe driving.

Enev *et al.* [55] analyzed driving behavior using in-vehicle sensor readings. The authors proposed a multi-class classification approach employing pairwise classifiers for each driver. The pairwise classifiers, utilizing binary classification models such as SVM, random forest (RF), KNN, and Naive Bayes, were trained on features extracted from sensor readings. These readings were recorded from driving on a closed loop setting in a parking lot and an open loop setting. Li *et al.* [56] proposed a driver identification approach that collects data from a 3-axis accelerometer, which records the lateral, longitudinal, and vertical accelerations for drivers' driving behavior analysis. The proposed approach contains a multi-modal ensemble algorithm comprising multiple classifiers, such as KNN, RF, MLP, and AdaBoost algorithms, that help identify the driver based on combining the output of the multi-classifiers. Ravi *et al.* [57] leveraged the sensor data collected from the CAN bus, which offers insights into driving behavior, and employed an LSTM deep learning model to distinctly identify the legitimate driver of a vehicle based on this data. Martinelli *et al.* [58] proposed a framework that compares the accuracy and effectiveness of driver detection based on CAN bus data with several machine learning algorithms, including J48, J48graft, J48consolidated, RandomTree, and RepTree. Yang *et al.* [59] proposed a deep learning architecture (Driver2vec) to identify drivers. Driver2vec is a custom deep learning model that leverages performance gains of temporal convolutional networks, embedding separation power of triplet loss and classification accuracy of gradient boosting decision trees to ensure precise driver identification accuracy. However, these papers do not analyze drivers' behavior towards unsafe driving detection.

VIII. Conclusion

In this paper, we present AutoWatch, an efficient approach that utilizes graph-based techniques for learning relationships among diverse data samples, with embeddings that can be used for subsequent prediction tasks. AutoWatch integrates techniques such as PCA, cosine similarity, HVG, and GNN to establish meaningful connections between data points and generate embeddings. These embeddings are subsequently employed in an MLP classifier for tasks related to driver identification and detection of unsafe driving maneuvers. Through empirical studies on two datasets, we demonstrate that AutoWatch significantly enhances classification performance compared to baseline methods. Furthermore, our analysis highlights the impact of AutoWatch on the embeddings derived from data samples. In addition, we emphasize the importance of the number of connections in influencing AutoWatch's performance. AutoWatch provides a robust framework for representing data samples as graphs, offering a versatile foundation for diverse classification tasks.

References

[1] NHTSA, "Vehicle theft prevention," *United States Department of Transportation*, Accessed October 18, 2023, https://www.nhtsa.gov/road-safety/vehicle-theft-prevention.

[2] ——, "Federal motor vehicle theft prevention standard," *United States Department of Transportation*, Accessed October 18, 2023, https://www.federalregister.gov/documents/2002/06/26/02-15903/federal-motor-vehicle-theft-prevention-standard.

[3] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès, "Lock it and still lose it—on the ({In) Security} of automotive remote keyless entry systems," in *25th USENIX security symposium (USENIX Security 16)*, 2016.

[4] K. Tindell, "Can injection: Exploring the possibilities," https://kentindell.github.io/2023/04/03/can-injection/, 2023, accessed: December 25, 2023.

[5] M. D. Pesé and K. G. Shin, "Survey of automotive privacy regulations and privacy-related attacks," 2019.

[6] F. T. Commission, "Connected cars: Privacy and security issues related to connected and automated vehicles," https://www.ftc.gov/news-events/events/2017/06/connected-cars-privacy-security-issues-related-connected-automated-vehicles, 2017, accessed: December 25, 2023.

[7] WHO, "Road traffic injuries," *World Health Organization*, June 20, 2022 (Accessed October 18, 2023), https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[8] IIHS, "Fatality facts 2021 yearly snapshot," *Insurance Institute for Highway Safety - Highway Loss Data Institute*, =Accessed October 18, 2023, https://www.iihs.org/topics/fatality-statistics/detail/yearly-snapshot.

[9] NHTSA, "Fatality analysis reporting system," *United States Department of Transportation*, =Accessed October 18, 2023, https://www.nhtsa.gov/crash-data-systems/fatality-analysis-reporting-system.

[10] E. K. Adanu, D. Brown, S. Jones, and A. Parrish, "How did the covid-19 pandemic affect road crashes and crash outcomes in alabama?" *Accident Analysis & Prevention*, vol. 163, p. 106428, 2021.

[11] C. Tingvall and N. Haworth, "Vision zero-an ethical approach to safety and mobility," in *6th ITE international conference road safety & traffic enforcement: Beyond 2000*, 1999.

[12] P. Agbaje, A. Anjum, A. Mitra, E. Oseghale, G. Bloom, and H. Olufowobi, "Survey of interoperability challenges in the internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22 838–22 861, 2022.

[13] M. H. Shahriar, W. Lou, and Y. T. Hou, "Cantropy: Time series feature extraction-based intrusion detection systems for controller area networks."

[14] P. Agbaje, A. Anjum, A. Mitra, G. Bloom, and H. Olufowobi, "A framework for consistent and repeatable controller area network ids evaluation," in *Fourth International Workshop on Automotive and Autonomous Vehicle Security*, 2022.

[15] J. Xu, S. Pan, P. Z. Sun, S. H. Park, and K. Guo, "Human-factors-in-driving-loop: Driver identification and verification via a deep learning approach using psychological behavioral data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3383–3394, 2022.

[16] B. I. Kwak, J. Woo, and H. K. Kim, "Know your master: Driver profiling-based anti-theft method," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2016, pp. 211–218.

[17] "Machine learning techniques to identify unsafe driving behavior by means of in-vehicle sensor data," *Expert Systems with Applications*, vol. 176, p. 114818, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421002591

[18] S. Zhang, O. T. Ajayi, and Y. Cheng, "A self-supervised learning approach for accelerating wireless network optimization," *IEEE Transactions on Vehicular Technology*, 2023.

[19] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, and T. Baker, "Analysis of dimensionality reduction techniques on big data," *Ieee Access*, vol. 8, pp. 54 776–54 788, 2020.

[20] O. T. Ajayi, X. Cao, H. Shan, and Y. Cheng, "Self-renewal machine learning approach for fast wireless network optimization," in *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2023, pp. 134–142.

[21] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d'Enza, A. Markos, and E. Tuzhilina, "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.

[22] A. Ghosal, A. Nandy, A. K. Das, S. Goswami, and M. Panday, "A short review on different clustering techniques and their applications," *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, pp. 69–83, 2020.

[23] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[24] L. Lacasa, A. Nunez, É. Roldán, J. M. Parrondo, and B. Luque, "Time series irreversibility: a visibility graph approach," *The European Physical Journal B*, vol. 85, pp. 1–11, 2012.

[25] L. Lacasa, B. Luque, F. Ballesteros, J. Luque, and J. C. Nuno, "From time series to complex networks: The visibility graph," *Proceedings of the National Academy of Sciences*, vol. 105, no. 13, pp. 4972–4975, 2008.

[26] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[27] A. d'Aspremont, L. Ghaoui, M. Jordan, and G. Lanckriet, "A direct formulation for sparse pca using semidefinite programming," *Advances in neural information processing systems*, vol. 17, 2004.

[28] P. Tian, W. Liao, W. Yu, and E. Blasch, "Wscc: A weight-similarity-based client clustering approach for non-iid federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20 243–20 256, 2022.

[29] P. Agbaje, A. Anjum, Z. Talukder, M. Islam, E. Nwafor, and H. Olufowobi, "Fedcime: An efficient federated learning approach for clients in mobile edge computing," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 2023, pp. 215–220.

[30] B. Luque, L. Lacasa, F. Ballesteros, and J. Luque, "Horizontal visibility graphs: Exact results for random time series," *Physical Review E*, vol. 80, no. 4, p. 046103, 2009.

[31] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[32] J. Liu, G. P. Ong, and X. Chen, "Graphsage-based traffic speed forecasting for segment network with sparse data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1755–1766, 2020.

[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[34] A. S. Yuksel and Atmaca, "Driving behavior dataset," https://doi.org/10.17632/jj3tw8kj6h.2, 2020, mendeley Data, V2.

[35] M. N. Azadani and A. Boukerche, "Driving behavior analysis guidelines for intelligent transportation systems," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 7, pp. 6027–6045, 2021.

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

[37] Y. Yao, X. Zhao, H. Du, Y. Zhang, G. Zhang, and J. Rong, "Classification of fatigued and drunk driving based on decision tree methods: a simulator study," *International journal of environmental research and public health*, vol. 16, no. 11, p. 1935, 2019.

[38] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[39] J. Wei, C. Long, J. Li, and J. Zhao, "An intrusion detection algorithm based on bag representation with ensemble support vector machine in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 24, p. e5922, 2020.

[40] L. Rokach and O. Maimon, "Decision trees," *Data mining and knowledge discovery handbook*, pp. 165–192, 2005.

[41] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *International Journal of Advanced Networking and Applications*, vol. 7, no. 4, p. 2828, 2016.

[42] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[43] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in *2019 international conference on intelligent computing and control systems (ICCS)*. IEEE, 2019, pp. 1255–1260.

[44] G. Baldini and D. Geneiatakis, "A performance evaluation on distance measures in knn for mobile malware detection," in *2019 6th international conference on control, decision and information technologies (CoDIT)*. IEEE, 2019, pp. 193–198.

[45] L. Noriega, "Multilayer perceptron tutorial," *School of Computing. Staffordshire University*, vol. 4, no. 5, p. 444, 2005.

[46] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayer perceptron," *Journal of neural network computing*, vol. 2, no. 2, pp. 40–48, 1990.

[47] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[48] E. Martin, "Determining which drivers must be listed on a car insurance policy," https://www.insure.com/car-insurance/determining-which-drivers-must-be-listed-car-insurance-policy.html, accessed: December 6, 2023.

[49] S. Arumugam and R. Bhargavi, "A survey on driving behavior analysis in usage based insurance using big data," *Journal of Big Data*, vol. 6, pp. 1–21, 2019.

[50] C. K. Joshi, F. Liu, X. Xun, J. Lin, and C. S. Foo, "On representation knowledge distillation for graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[51] H. He, J. Wang, Z. Zhang, and F. Wu, "Compressing deep graph neural networks via adversarial knowledge distillation," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 534–544.

[52] E. Khosravi, A. M. A. Hemmatyar, M. J. Siavoshani, and B. Moshiri, "Safe deep driving behavior detection (s3d)," *IEEE Access*, vol. 10, pp. 113 827–113 838, 2022.

[53] P. Khosravinia, T. Perumal, and J. Zarrin, "Enhancing road safety through accurate detection of hazardous driving behaviors with graph convolutional recurrent networks," *IEEE Access*, vol. 11, pp. 52 983–52 995, 2023.

[54] Y. Ma, Z. Xie, S. Chen, F. Qiao, and Z. Li, "Real-time detection of abnormal driving behavior based on long short-term memory network and regression residuals," *Transportation research part C: emerging technologies*, vol. 146, p. 103983, 2023.

[55] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting." *Proc. Priv. Enhancing Technol.*, vol. 2016, no. 1, pp. 34–50, 2016.

[56] Z. Li, K. Zhang, B. Chen, Y. Dong, and L. Zhang, "Driver identification in intelligent vehicle systems using machine learning algorithms," *IET Intelligent Transport Systems*, vol. 13, no. 1, pp. 40–47, 2019.

[57] C. Ravi, A. Tigga, G. T. Reddy, S. Hakak, and M. Alazab, "Driver identification using optimized deep learning model in smart transportation," *ACM Transactions on Internet Technology*, vol. 22, no. 4, pp. 1–17, 2022.

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

[59] J. Yang, R. Zhao, M. Zhu, D. Hallac, J. Sodnik, and J. Leskovec, "Driver2vec: Driver identification from automotive data. arxiv 2021," *arXiv preprint arXiv:2102.05234*.