A Mobile Telematics Pattern Recognition Framework for Driving Behavior Extraction

Mohammad Siami, Mohsen Naderpour, *Member, IEEE*, Jie Lu, *Fellow, IEEE*.

***Abstract*— Mobile telematics is a relatively new innovation that involves collecting data on driving behavior using the internal sensors in a smartphone rather than from an in-vehicle data recorder. However, telematics data are usually not labeled, which makes extracting driving patterns from them very difficult. Therefore, unsupervised learning algorithms play an important role in this field. In addition, most current research is based on datasets developed in a laboratory or from site investigations and questionnaires, which are very different from real-world driving behaviors. To advance unsupervised learning techniques in this field, and to fill the gap in findings based on real-world data, we have developed an unsupervised pattern recognition framework for mobile telematics data. The framework comprises three main components: a self-organizing map, a nine-layers deep auto-encoder, and partitive clustering algorithms. The SOM algorithm reduces the complexity of the data, the deep auto-encoder extracts the features, and the clustering algorithm groups driving events with similar patterns into behaviors. Further, given clustering with mobile telematics data is an under-researched area, we undertook an empirical comparison of five well-known clustering algorithms to determine the strengths and weaknesses of each method and which is best suited to categorizing driving styles. The study was conducted with a real-world insurance dataset containing 500,000 journeys by 2500 drivers, and the results were evaluated against three measures – Davis Boulding, Calinski Harabasz, and execution time. Overall, we find that k-means clustering and a self-organizing map were able to extract more accurate patterns than others. A statistical analysis of the 29 clusters produced by SOM and k-means, revealed 29 unique driving styles, all of which can be found in the transportation literature. The results from the study, with support from the corresponding literature review, demonstrate the efficacy of the presented framework in unsupervised settings. Additionally, the results provide a basis for developing a future risk analysis and automatic decision support system for usage-based insurance companies.**

Index Terms— mobile telematics, pattern recognition, vehicle driving, unsupervised learning

I. Introduction

TECHNOLOGICAL improvements in the internet of things (IoT) have led to a wide range of applications that enhance our lives, including smart homes, healthcare systems, vehicle monitoring, and greater awareness of environmental problems. IoT applications have two main advantages. First, they allow hardware devices to connect with their surrounding environment and each other to report on or accomplish a task. Second, they generate huge amounts of data that are useful for behavioral and environmental analytics. Further, growth in the use of smartphones, as one type of interconnected device, is likely to further increase the number of useful IoT applications developed in future years [4].

Telematics is one such IoT application, which involves integrating sensors, computer systems, and communications to gather information about a vehicle's operations. However, using this technology requires different kinds of velocity and acceleration sensors to be installed in the vehicle, which are expensive and hard to develop. To overcome this problem, Malalur et al. [5] invented a new kind of telematics, known as mobile telematics, that uses the sensors in smartphones to record and track driving behavior. Because most people already own a smartphone, mobile telematics offers a new, low-cost alternative for collecting data about driving behavior [6].

All smartphones contain at least one instrument capable of measuring position by connecting to a fixed communication system, such as a cellular radio station, wifi access point, or GPS receiver. Smartphones can also contain a three-axis accelerometer, a gyroscope, and/or a compass. These internal sensors give mobile telematics apps a wide scope to gather driving style data. The apps are easy to use, and the initial hardware cost is either very low or free if the user already has a smartphone [7]. Further, the massive amounts of data they collect benefit a range of analytical uses like road safety [8], intelligent transportation systems [9], usage-based insurance [10], and others. Perhaps more importantly, these apps can help people assess and improve their own driving behavior by providing feedback on their driving styles with incentives to change bad habits [5]. Thus, it is unsurprising that one of the biggest beneficiaries of mobile telematics is the insurance industry. With mobile telematics apps, insurers no longer need to rely on expensive in-vehicle sensor installations to take advantage of driver "monitoring". As a result, many insurers are specifically targeting drivers that are willing to use mobile telematics with their marketing campaigns [7].

All these benefits, however, are predicated on good definitions of driving. Thus, driving behavior detection methods typically fall into two main groups [4]. The first is rules-based detection, which identifies risky habits by defining different thresholds for dangerous and normal behavior [11]. The rules and thresholds are usually developed by transportation experts in autonomous driving, driving

The authors are with the Decision Systems and e-Service Intelligence Laboratory, Center for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: mohammad.siaminamini@uts.edu.au; Mohsen.naderpour@uts.edu.au; jie.lu@uts.edu.au).

simulation, behavioral risk assessment, and similar fields [12]. The second approach again relies on transportation experts, this time with a set of predefined templates that describe different driving styles ranging from normal to dangerous. A set of pattern matching algorithms and machine learning models are then used to classify a driver's behavior according to the most similar patterns [4]. Yet, developing good definitions of something so fluid and dynamic as driving behavior is difficult, even for experts.

Further, although extensive research has been undertaken on driving style analytics with supervised learning algorithms, only a few researchers have studied unsupervised learning methods. Moreover, much of the research up to now has been conducted using data collected from questionnaires, site investigations, or laboratory simulations. However, driving behavior in the real world is completely different from the simulated behavior in generated data. We believe the dynamic properties of human behavior mean that simulated data cannot reflect all driving habits. To the best of our knowledge, Lee and Jang [13] are two of a few researchers to have developed a framework for driving style pattern recognition using real-world data and unsupervised learning techniques. However, their framework is based on collecting data with in-vehicle recorders, which are expensive and difficult to implement. Moreover, their research does not sufficiently consider the most relevant state-of-the-art techniques in driving style analytics, such as deep learning and auto-encoders.

Hence, our goal with this research was to develop an up-to-the-minute framework for driving style pattern recognition based on the cheaper, easier alternative of mobile telematics, and one that leverages the current state-of-the-art in machine learning. That said, even though mobile telematics holds a great deal of promise, there are several challenges to overcome. The lack of research on unsupervised learning algorithms is the first obstacle. The second involves clustering and the disparate findings in the literature about which algorithms are the most effective. Moreover, as a field, we need to determine which clustering algorithms are able to classify dynamic driving habits without pigeon-holing drivers into a set of predefined templates or rules. These challenges led to the following four research questions:

1) How can raw mobile telematics data that only contain geographic position features be used to categorize driving patterns?
2) How can the framework benefit from the current state-of-the-art techniques in unsupervised learning, such as deep auto-encoders?
3) What is the best clustering algorithm for categorizing driving styles with unlabeled mobile telematics data?
4) Do the driving behaviors identified by the best unsupervised learning algorithms and optimal clustering algorithms correlate to known normal or abnormal driving styles?

The framework we developed is based on unsupervised learning models that transforms mobile telematics data into data streams of driving characteristics. A change detection algorithm identifies the most significant and informative time windows in the stream about the decisions drivers have made, after which the data is ready for clustering. A self-organizing map then reduces the complexity of the data, and a deep auto-encoder extracts the features.

To answer research questions 3 and 4, we undertook an empirical study of five of the most well-known and commonly-used partitional clustering algorithms in the field of pattern recognition to reveal the strengths and weaknesses of each, and to determine whether there is one best choice overall for categorizing driving styles from mobile telematics data. Clustering performance was measured against three different metrics – Davis Boulding, Calinski Harabasz, and execution time – and the data used was from a real-world insurance dataset of 500,000 driving trips by 2500 drivers.

In summary, the main contributions of this paper include:

- An approach for identifying the driving events with the highest rate of change according to three key characteristics – velocity, x-axis acceleration, and y-axis acceleration – using relative unconstrained least-squares importance fitting (RuLSIF).
- A novel unsupervised learning framework specifically designed for mobile telematics data that uses significant patterns in lieu of labels to identify driving behaviors for clustering.
- A self-organizing map for reducing the complexity of data, and A deep auto-encoder architecture with nine layers that automatically extracts features from driving characteristics, are used to prepare data for partitive clustering.
- An empirical assessment of five partitive clustering algorithms on SOM and DAE results by the Davis Boulding and Calinski Harabasz indexes as well as execution time. The performance results show that a self-organizing map and k-means clustering are the best combination of two-stage clustering clustering similar driving patterns into a set of driving behaviors.
- Verification that the categories extracted correlate to known driving styles in the transportation literature.

The rest of this paper is organized as follows. The background of the work has been explained in Section II. Section III contains a review of the literature that informed this research. Section IV presents the details of the mobile telematics pattern recognition framework. Section V contains the implementation and experimental results. Section VI describes the various driving patterns extracted by SOM and k-means. Finally, Section VII concludes the paper and describes future work.

II. Background

This section provides the background of this study, including a brief summary of the change detection algorithm, the self-organizing map algorithm, the deep auto-encoder, and the partitive clustering methods used in this paper.

A. Change Detection Algorithm

The change detection algorithm is an algorithm to find time windows of major change within time-series data – most

commonly through statistical techniques. Change detection algorithms have a wide range of applications, e.g., signal segmentation [14], climate change detection [15], and driving behavior analytics [13].

Let consider $\mathcal{Y}(t) \in R_d$ as time-series data with d dimensions at time t, and $y(t) = [Y(t)^T, Y(t+1)^T, \dots, Y(t+k-1)^T]^T \in R_{dk}$ is a consecutive time window of length k at time t. Following Liu et al. [16] strategy, the dissimilarity between $y(t)$ and $y(t+n)$ is calculated from the equation below, and the result is used as a change score to reflect the amount of change between two time windows.

$$ChangeScore = D(p_t||p_{t+n}) + D(p_{t+n}||p_t) \quad (1)$$

where p_t and p_{t+n} are the probability distributions of $\mathcal{y}(t)$ and $\mathcal{y}(t+n)$. For simplicity, hereafter, we denote this dissimilarity as $D(p||p')$ instead of $D(p_t||p_{t+n})$.

To calculate the dissimilarity measure between two different time segments, Liu et al. [16] proposed the relative unconstrained least-squares importance fitting (RuLSIF) algorithm. RuLSIF. It calculates the change between two consecutive time windows with a density-based dissimilarity measure:

$$D(p||p') = -\frac{\alpha}{2n}\sum_{i=1}^{n}\hat{g}(Y_i)^2 - \frac{1-\alpha}{2n}\sum_{i=1}^{n}\hat{g}(Y_i')^2 + \frac{1}{n}\sum_{i=1}^{n}\hat{g}(Y_i) - \frac{1}{2} \quad (2)$$

where n is the window size, and Y_i *and* Y_i' are two consecutive time windows in d-dimensional time-series data. $\hat{g}$ is the density-ratio estimation of the data samples, and α is a constant variable.

B. Self-organizing Map

Self-organizing map (SOM) is a special type of unsupervised learning algorithm that generate a discretized map of an input space. SOMs have become a common technique in a wide range of applications, such as data visualization, dimension reduction, and vector quantization [17]. The main advantage of SOM is that they reduce computation costs, which is particularly valuable if clustering is part of one's strategy. Given the complexity of calculating distances within multi-dimensional data, most clustering algorithms are computationally greedy, even with a small number of records. SOM decrease computation costs by abstracting a prototype of the input data. A clustering algorithm can then be used to classify the abstracted data instead of the full dataset [18]. Another advantage of SOM is its ability to tolerate noise. Each node in a SOM represents a group of input data, so it is less sensitive to data generated in noisy environments [19]. In contrast, one of the greatest weaknesses of SOMs is detecting outliers. By definition, outliers are rare data points and, therefore, SOMs have difficulty generating a suitable prototype to represent those data [20].

The gist of these algorithms is to map the input data into a topographical map with *N* nodes on a regular two-dimensional rectangular or hexagonal grid, where each node has d number of features with a weight $\omega_i = [\omega_{i1}, \omega_{i2}, \dots, \omega_{id}]^T$. The algorithm is iterative. In each iteration step *t*, a data sample *x*(*t*) is randomly selected from the training data, and the distances are calculated between *x*(*t*) and all the nodes. The most similar node to *x*(*t*) is selected with

$$c = argmin(dist(x(t), \omega_i), \quad \forall\, i\ in\ [1,2,\dots,N] \quad (3)$$

where $dist(x(t), \omega_i)$ is equal to the distance of the sample x(t) with the i[th] node.

After a "winning" neuron has been selected, it and its neighboring neurons are updated with a weight updating rule:

$$\omega_k(t+1) = \begin{cases} \omega_k(t) + \gamma(t)h_{kc}(t).\left(x(t) - \omega_j(t)\right), \forall k \in N_c \\ \omega_k(t), \quad else \end{cases} \quad (4)$$

where N_c is the winning neuron's neighbors, and $\gamma(t)$ is the learning rate, which is reduced in each iteration (t) with the following equation:

$$\gamma(t) = \gamma_0.exp\left(-\alpha.\frac{t}{\tau}\right) \quad (5)$$

where γ_0 is the initial learning rate, α is the exponential decaying constant, and τ is the maximum number of iterations. $h_{kc}(t)$ is a neighborhood kernel function that indicates the distance of the k[th] neuron to the winning neuron c, as calculated by

$$h_{kc} = \exp\left(-\frac{[(x_k-x_c)^2+(y_k-y_c)^2]}{2(\sigma(t)^2)}\right) \quad (6)$$

where $\sigma(t)$ is equal to the width of the neighborhood function and decreases in each iteration t by

$$\sigma(t) = \gamma_0.\exp\left(-\frac{t}{\tau}.\log(\sigma_0)\right) \quad (7)$$

where σ_0 is the initial width [21].

C. Deep Auto-encoder

A deep auto-encoder model is a group of several auto-encoders that are arranged in a neural network architecture. A simple auto-encoder has two parts, an encoder and a decoder. An example of a deep auto-encoder is shown in Fig. 1.

In the encoding layer, the encoder function $h = f(Wx + b)$ is used for each layer to encode the input data. The encoding stage continues up to the middle layer, at which point a decoder function $h = f(W'x + b')$ begins to reconstruct the encoded input data. Sigmoid, tanh, soft sign, and Relu functions are the most prominent activation functions for encoder and decoder functions [22].

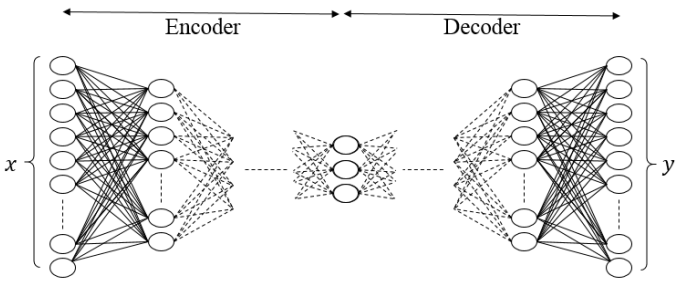

Fig. 1. A deep auto-encoder with many layers

The set of parameters for a basic auto-encoder comprises $W_l, W'_{l'}, b_l, b'_{l'}$. These parameters are trained to minimize the

loss function by

$$loss = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2 \quad (8)$$

The training procedure is unsupervised, and the middle layer represents the encoded version of input data [22]. In this paper, we used a deep auto-encoder to automatically extract the features from the driving style data.

D. Partitive Clustering

Partitive clustering is an unsupervised learning technique that clusters unlabeled input data into a number of partitions, i.e., members are grouped according to distance-based similarity. Partitive clustering algorithms assume that the input data can be categorized into prototypes; thus, they are also known as prototype-based clustering algorithms. The main goal is to compress the data into these prototypes. Each partitive clustering algorithm has different methods of defining the prototypes for the input data. For example, one of the most famous partitive clustering algorithms, k-means, uses the K-means++ algorithm to find the initial prototypes [23]. Partitive clustering algorithms have been used in a wide range of applications, from big data clustering [24] for customer segmentation [25, 26], to weather prediction [27], to biomedical health [28], and many others. The main steps of a partitive clustering algorithm are outlined in Algorithm 1 below.

Algorithm 1: Partitive clustering algorithm [23]

Input: Dataset and K number of prototypes, M max iteration
Output: data points with a cluster label

1. ***Initialize K data points from the input data as initial cluster prototypes.***
2. ***Assign each data point to the closest prototype using a distance function.***
3. ***Recalculate the center of each cluster with these new data points.***
4. ***Repeat steps two and three if the clusters do not change significantly.***

III. Literature Review

The following sub-sections summarize relevant literature relating to smartphone-based vehicle telematics, driving style analytics, and pattern recognition. The review also includes a brief summary of the self-organizing map algorithm, the change detection algorithm, and the clustering methods used in this paper.

A. Mobile Telematics

Smartphones provide the capabilities of a home computer system in a mobile environment. Further, all smartphones have at least one internal measurement sensor and the ability to connect and transfer data to a remote server. Widespread growth in the use of smartphones means these devices have become a critical part of collecting data for industries like insurance and transportation companies that have highly mobile customers of all ages and income levels [4]. For example, prior to mobile telematics, an insurance company wanting to take advantage of driver telemetry had to rely on an in-vehicle data recorder. One insurance company in Italy who uses telematics devices for data gathering explains that a telematics business model has wide-ranging benefits for all stakeholders: customers, partners, technology providers, society, and, of course, themselves. Obviously, increasing profit is the main benefit for the insurer. By proactively selecting good drivers according to their risk level, the profitability of this Italian insurer increased by 30% even though they were offering 5% to 30% discounts on premiums [29].

However, using an in-vehicle data recorder to gather telematics data is very expensive and difficult to implement. Hence, once smartphones emerged as a phenomenon, mobile telematics technology soon followed. Initially, the reliability of this new technology was a major question. But, in 2014, Handel et al. [30] conducted an empirical comparison of the data generated by a smartphone versus a traditional in-vehicle data recorder. In four main categories of reliability (accuracy, availability, integrity, and continuity of service) the authors found that smartphones could be a valid and appropriate tool for collecting telematics data.

B. Driving Style Analytics

Wahlström et al. [4] divided the practical applications of mobile telematics into seven categories: navigation, transportation mode classification, cooperative intelligent transportation systems, mobile cloud computing, driver behavior classification, and monitoring road conditions. Our focus is on driving style analytics and, within this, driver behavior classification and pattern recognition.

According to the study of Wahlström et al. [4], driving behavior classification methods typically follow one of two approaches. The first is to define driving behavior according to one or more thresholds. For example, "safe acceleration" might be defined as when the norm of acceleration or deceleration is less than 2 m/s^2; velocity changes beyond this threshold would be classified as extreme events [2]. The second approach is to define a range of templates that represent different driving behaviors. For example, a harsh cornering event might be defined in a template by an acceleration value on the x and/or y-axis during a specified time window. Harsh cornering events in drivers are then identified by calculating the similarity of their behavior to the template definition.

However, technological advancements, and particularly the integration of machine learning into pattern matching algorithms, are now providing opportunities to classify driving styles more acutely than ever before [31, 32]. For instance, Wang and Xi [33] proposed a binary classification solution to distinguish aggressive driving patterns from moderate ones. Their method involves support vector machine (SVM) and k-means clustering to decrease execution times and improve prediction accuracy. The k-means clustering algorithm first reduces the complexity of the input data, then SVM distinguishes between normal and abnormal driving styles. Cross-validation experiments show the approach to be faster and more accurate than SVM alone. In another study, Henriksson [34] introduced a pattern recognition framework to

identify driving contexts from vehicle-generated data. City driving styles were compared to open road driving by finding the hidden relations between driving attributes in these two contexts. In a comparison between SVM and a hidden Markov model, the results show SVM to be more reliable.

Through a driving behavior monitoring system, Yu et al. [3] categorized different, unusual driving behaviors into six groups: weaving, swerving, sideslipping, fast U-turns, turning with a wide radius, and sudden braking. Their method not only distinguishes between normal and abnormal driving patterns but also specifies the type of dangerous driving behavior. In a comparison between SVM and a neural network as a training algorithm for the classification model, the neural network model was better able to detect dangerous driving patterns.

Driver identification is another research area in driving style analytics. To date, researchers have applied several artificial intelligence and machine learning algorithms to identify who is behind the wheel. A data transformation method was proposed by Dong et al. [35] to transform trajectory data into information that is usable in deep learning. They used a convolutional neural network (CNN) and a recurrent neural network (RNN) to distinguish drivers from passengers in a real-world dataset collected by a European insurance company. In a subsequent study, Dong et al. [36] proposed another model based on an auto-encoder regularized network (ARNet) to estimate the total number of drivers using one vehicle. The algorithm contains multiple levels of neural networks, including a gated recurrent unit (GRU), an auto-encoder, and logits. Insurance companies can take particular advantage of these models because underwriters are very interested in how many people are actually driving a car, especially when policies and premiums are linked to the age and number of drivers. A Mobile telematics data analytics framework based on supervised learning method has been proposed by Siami et al. [37], they detected the gender of drivers from the driving styles using Choquet Fuzzy Integral Vertical Bagging Random Forest Classifier. In another study, a driver identification methodology was proposed by Moreira-Matias and Farah [38], using trip-based historical datasets collected by in-vehicle data recorders to identify the category of driver behind the wheel. They took the advantage of the driver-labelled trip data to build a pattern of different drivers in different categories using various supervised learning algorithms.

The above methods are all supervised learning techniques that have shown outstanding performance in comparison to traditional methods of driving analytics. In fact, most current studies on driving style analytics with machine learning techniques are conducted in supervised learning scenarios. Only a few consider unsupervised methodologies for driving style analytics. One study, by Liu et al. [39], maps driving style patterns into three-dimensional data so as to visualize each pattern as a different color. A deep auto-encoder framework reduces the data streams into three-dimensional data. Each dimension is then mapped to either red, green, or blue – one color for each unique behavior – and the auto-encoder extracts the features from the behaviors. However, using their framework in real-world scenarios is somewhat challenging because they used synthetic data to train the deep learning model. In the real world, many data are uncharacteristic and completely different from the data generated in a laboratory. Lee and Jang [13] also proposed an unsupervised learning framework to characterize driving style patterns, this time with data generated by in-vehicle data recorders. However, their study did not extend to exploring the performance of different clustering algorithms for driving style extraction. Moreover, the correlation between their results and driving styles described in the literature was not fully investigated. These issues, combined with the problem of in-vehicle data, warrant further study in a mobile telematics setting. Shouno [40] incorporated a variational auto-encoder into a deep unsupervised learning framework for the purposes of reducing the input dimensions down to a two-dimensional space. Driving styles were then characterized according to a topological map. He tested his framework on a Honda Driving simulator with 59 drivers, which, again is simulated data and completely different from those found in the real-world.

Driving behavior is complex, nuanced, and dynamic, and understanding driving behavior with synthetic data which everything rely on good definition cannot show the behavior of drivers in the real-world. In addition, the lack of labeled data is a challenge, as highlighted in [41]. To the best of our knowledge, much of the research until now has been conducted on data gathered from either questionnaires, site investigations, or laboratory simulations. We believe that the dynamic properties of human behavior cannot be fully reflected in simulated data. Moreover, our literature review shows that driving style pattern recognition using mobile telematics data has not been studied in any great detail. This study is intended to fill those gaps.

IV. MOBILE TELEMATICS PATTERN RECOGNITION FRAMEWORK

The unsupervised learning framework presented in this paper is designed to extract driving patterns from trajectory data. Fig. 2. illustrates the two major steps in the framework. The first step is data preparation to prepare the trajectory data for analytics. Here, trajectories are transformed into streaming data to reveal driving behaviors, and change detection is applied to find the most important events in the raw telemetry. The second step involves categorizing the different driving behaviors using a two-stage clustering procedure.

A. Data Preparation

Data cleaning and preparation is an essential step for any data mining and knowledge discovery project. Hence, the primary goal of this step is to clean the data and reduce its complexity. There are two parts to this process: data transformation and change detection.

1) *Data transformation*

Smartphones record information about the position of a vehicle as geolocation coordinates, e.g., latitude and longitude, which can be used to locate a vehicle on a map or as a starting

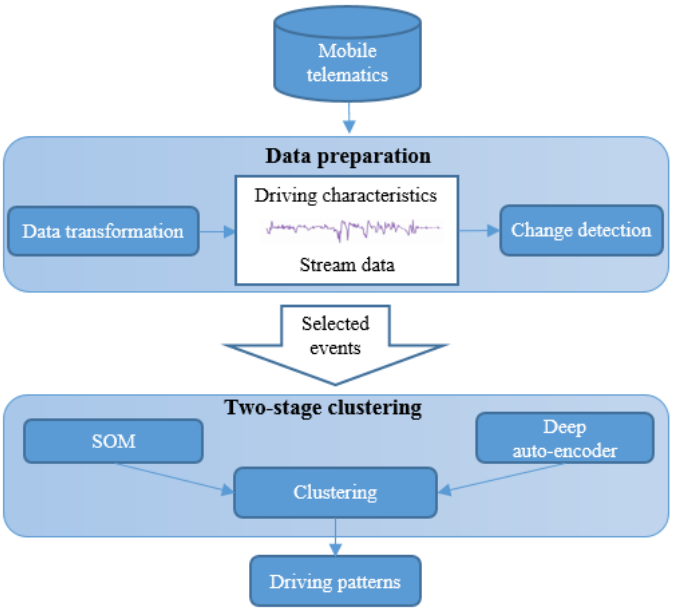

Fig. 2- The driving style pattern detection framework

point for estimating speed and acceleration. However, to be useful in the current application, the raw geolocation data needs to be processed. Some preliminary definitions used in this process follow.

Definition 1. The position of a vehicle in a 2D coordinate space at time t is P_t. A driver starts each trip at time 0 from location $P_0 = [0, 0]$ [35].

Definition 2. Mobile telematics devices generate a series of GPS data for each trip (tr) [42]

$$tr = P_0 \rightarrow P_1 \ldots \rightarrow P_i \ldots \rightarrow P_i$$

The starting point of a trip is P_0, and the end point is P_i. The trips associated with each driver occur at different times, day or night.

Definition 3. In line with international standard ISO 8855, we considered a two-dimensional coordinate system to calculate the driving characteristics. The coordinate system is depicted in Fig 3. The forward and backward directional movements of the car are plotted on the x-axis, and the left and right directional movements of the car are plotted on the y-axis. These assumptions are used to calculate the value of instantaneous velocity and acceleration. Therefore, changing the position of the vehicle in a forward or backward direction indicates x-axis movement and movement in a left or right direction indicates y-axis movement.

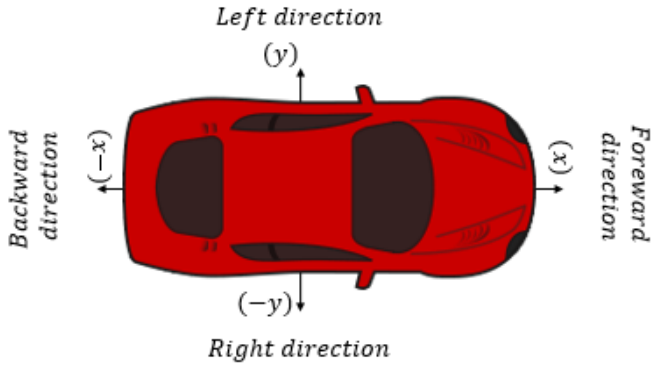

Fig. 3- Two-dimensional vehicle coordinate system (ISO 8855)

Definition 4. Let $|\overline{V_t}|$ be the instantaneous velocity of the vehicle at time t, calculated by

$$|\overline{V_t}| = \sqrt{\left(v_{x_t}\right)^2 + \left(v_{y_t}\right)^2} \quad (9)$$

where v_{x_t} and v_{y_t} show the instantaneous velocity of the vehicle at time t over the x and y axes. These are calculated by $v_{x_t} = \frac{\Delta x}{\Delta t}$ and $v_{y_t} = \frac{\Delta y}{\Delta t}$, respectively.

Definition5. $|\overline{A_t}|$ is the norm of an acceleration/deceleration event at time t, calculated by

$$|\overline{A_t}| = \sqrt{\left(a_{x_t}\right)^2 + \left(a_{y_t}\right)^2} \quad (10)$$

where a_{x_t} is the value of the instantaneous acceleration at time t over the x-axis, which is equal to $\frac{\Delta v_x}{\Delta t}$. Similarly, the instantaneous acceleration over the y-axis is $a_{y_t} = \frac{\Delta v_y}{\Delta t}$.

2) *Change detection*

The data streams generated from mobile telematics do not arrive ready for analysis as smartphones record data without any knowledge of the mechanical features of the vehicle [43]. For example, if a driver stops for a long time, these data would be recorded even though they are useless for our purposes. So, to ensure these types of data do not decrease the performance of the model, they must be identified and removed. We accomplish this through the change scores between time windows discussed in Section II.A, where each time window is associated with a driving event.

The change detection algorithm is formulated by considering $V(t)$, $A_x(t)$, and $A_y(t)$ as three dimensions which are velocity V(t), x-axis acceleration $A_x(t)$, and y-axis acceleration $A_y(t)$ respectively. Here, $V(t)$, $A_x(t)$, and $A_y(t)$ are three time windows with a length of k, which are:

$V(t) = [\mathcal{V}(t)^T, \mathcal{V}(t+1)^T, \ldots, \mathcal{V}(t+k-1)^T]^T \in velocity$;

$A_x(t) = [\mathcal{A}_x(t)^T, \mathcal{A}_x(t+1)^T, \ldots, \mathcal{A}_x(t+k-1)^T]^T \in x-axis\ acceleration$;

$A_y(t) = [\mathcal{A}_y(t)^T, \mathcal{A}_y(t+1)^T, \ldots, \mathcal{A}_y(t+k-1)^T]^T \in y-axis\ acceleration$, where T is the transpose. Let $Y(t) = [V(t), A_x(t), A_y(t)]$ is a the three-dimensional input data at time t, and $\mathcal{y}(t)$ be a group of n retrospective subsequences of input data at time t, which is:

$$\mathcal{y}(t) = [Y(t), \quad Y(t+1), \ldots, \quad Y(t+n-1)]$$

$\mathcal{y}(t)$ and $\mathcal{y}(t+n)$ are treated as two consecutive segments of the data stream. Fig. 4. illustrates an example of n retrospective consecutive segments of in one-dimensional time-series data [44]. The strategy is to calculate a dissimilarity score for these two segments using Eq. 1 as the measure of change. We selected RuLSIF as the change detection and scoring algorithm. RuLSIF is extremely good at detecting driving style changes [13], human activity sensing [16], and smart home signal processing [45] in data streams. RuLSIF calculates a change score using a density-based dissimilarity measure from

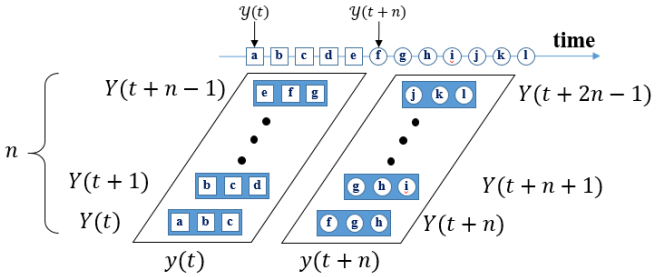

Fig. 4-. A sample of change detection with one-dimensional time-series data. $y(t)$ is the input data at time t, $Y(t)$ denotes k subsequences, and $y(t)$ is a group of retrospective subsequences. The value of n is equal to the window size, and k is the length of subsequence. In this example, k=3 three, but it is possible for this value to be bigger.

two consecutive time window by using Eq 2.

We used the implemented version of change detection algorithm developed by Liu et al. [16]. Three dimensions including velocity, x-axis acceleration and y-axis acceleration are the input data and the change score is the output.

B. *Two-stage Clustering*

Once the data has been prepared for analytics, a two-stage clustering algorithm categorizes the selected time windows into groups. In this stage, we used SOM and a deep auto-encoder to make a choice between them for subsequent clustering.

1) SOM

In our framework, the SOM is a lattice output space with a rectangular topology. The first step is to generate an initial SOM according to the number of input records. Then, one sample driving event (record) is selected randomly, and its similarity to all the rest of the SOM nodes is calculated in terms of Euclidean distance. The node with the smallest distance is selected as the best matching unit (BMU), and the selected sample is assigned to that. The winner and its neighboring nodes are then updated with the weight updating rule in Eq.4. The process continues until all data have been assigned to a corresponding node.

2) Deep Auto-encoder

The deep auto-encoder component consists of a number of neural networks with randomly generated weight and bias vectors, which are optimized during the training phase. Following Liu et al. [39], we designed a deep network with four encoder layers. The number of nodes in each layer are: 45 → 22 → 11 → 5 → 3 → 5 → 11 → 22 →45. Therefore, the network extracts the features by using the encoder layers 45 → 22 → 11 → 5 → 3. The gradient descent optimizer is used to minimize reconstruction errors.

The driving characteristics are denoted as $X \in \mathbb{R}^{D \times T}$, where D is the dimension of the input data, and T is the length of the time window. For example, a driving event at time (t) is:

$$X_t = (V_1, \dots, V_T, Ax_1, \dots, Ax_T, Ay_1, \dots, Ay_T)$$

The activation function is a hyperbolic tangent so the value of the input variables should be in the range $(-1,1)$. Obviously, the raw velocity and acceleration values will not fall within this range, so the data needs to be normalized before running the deep auto-encoder. We performed minimum and maximum normalization to transform the value of each dimension into $(-1,1)$.

The input data for the first layer of the deep auto-encoder is denoted as X_t, and the encoder function is

$$h_t^l = \tanh(\mathrm{W}_{en}^l X_t + b_{en}^l)$$

where W_{en}^l is a weight matrix for the encoder at the l^{th} layer and b_{en}^l is the bias vector for the l^{th} encoder layer.

The decoder function is a tanh function:

$$r_t^l = \tanh\left(\mathrm{W}_{de}^l h_t + b_{de}^l\right)$$

where W_{de}^l is the weight matrix and b_{de}^l is the bias vector.

An assumption during the encoder-decoder process is that the output value of r_t^l is equal to x_t^l. Thus, the objective function needs to calculate the reconstruction error between r_t^l and x_t^l:

$$O(V^l) = \frac{1}{N_V}\sum_{t=1}^{N_V} \left\| r_t^l - x_t^l \right\|_2^2 \quad (11)$$

where $\frac{1}{N_V}\sum_{t=1}^{N_V} \left\| r_t^l - x_t^l \right\|_2^2$ is the average value of the squared error between the reconstructed data and the input data.

A gradient descent optimizer with a learning rate of λ helps to minimize the reconstruction errors. Finding the best learning rate for a deep auto-encoder is typically very difficult and time-consuming. Hence, we have opted for a linear grid search, where the first learning rate considered is λ^* and the search distance is θ, which is very small. This results in a learning rate of $\lambda^+ = \lambda^* + \theta^+$. θ is updated with

$$\theta^+ = \begin{cases} \theta & O(V^l) \geq O^+(V^l) \\ -0.5 \times \theta & O(V^l) < O^+(V^l) \end{cases} \quad (12)$$

The search stops when the change in the construction error between $O(V^l)$ and $O^+(V^l)$ falls below a set threshold.

The features extracted through this deep auto-encoder process are then carried forward for use in the partitive clustering step.

3) Clustering

The SOM and deep auto-encoder algorithms have reduced the data to an abstract subspace. However, there will still be too many points to analyze directly, so they need to be clustered into similar groups. As mentioned in the Introduction, no research has been undertaken to determine whether there is a definitive best choice for clustering unlabelled telematics data. Therefore, we conducted our own empirical study on this issue with a range of different partitive clustering algorithms. Our hope is to not just find the most suitable choice for our framework, but to generate results that contribute to the debate on which clustering algorithm(s) might be best for mobile telematics data. The study is presented in Section IV.C.

V. Implementation and Data Analysis

We implemented our framework in Python 2.7 on an Intel® Xeon® 3.01 GHz CPU, 64 GB of RAM, and running a Linux operating system. The software platform was Anaconda 2.7. The specific implementations of the SOM library [46] and the RuLSIF library [16]. The Tensorflow is used for deep auto-encoder implementation.

A. *Data Preparation*

Our source data was a large-scale dataset collected by a European insurance company containing trip data for over

500,000 journeys from more than 2500 drivers. The computational cost to process this entire dataset would be extremely high. But, according to Dong et al. [35], each person has their own driving patterns so no new useful information would be gained by analyzing more than a few trips per driver. We selected the 20 longest trips per driver to include in the analysis. Thus, the final dataset contained 50,000 journeys (20 trips × 2500 drivers). Table I provides brief details about the data used.

TABLE I: SELECTED DATASET

Trips	Drivers	Journeys per driver	Traveling time (minute)		
			Min	Average	Max
50,000	2,500	20	23:21	26:13	30:00

To extract the driving characteristics from the trajectory data, we split the data into three streams. The first stream was velocity, containing the speed of a vehicle during a trip at any given time. The second and third streams were acceleration over the x and y axes. These streams were used to assess hard breaking, sharp starts [13], and cornering behavior [2, 30].

To remove useless data, we divided the data into time windows of approximately 15 seconds with a slide in steps of 1 second because, according to Zhang et al. [47], it takes at least 15 seconds to complete a single driving event. The RuLSIF-based change detection scores were then calculated for each time window. Fig. 5 shows the input and output of the RuLSIF change detection algorithm with velocity, x-acceleration, and y-acceleration as the three input variables. Fig. 5. shows the change score for the corresponding time frame. Approximately 7.9 million time windows were assessed. Following Lee and Jang [13], we selected the 5% with the highest RuLSIF scores to represent the most significant changes, and further selected all windows with a change score greater than a threshold of 68.598. This left 394,833 windows, each representing one driving event with 15 seconds of data.

B. *Two-stage Clustering*

After preparing the data and selecting the time windows with highest change score, the events were ready to analyze for their driving characteristics. As mentioned, we used SOM to reduce the complexity of data and deep auto-encoder to extract the features. In SOM, defining a map with an appropriate number of nodes is crucial because, when n is small, the prototype will be very generic and, when n is very large, the prototype will be very detailed. Therefore, to define an optimal number of nodes, we followed Céréghino and Park [48] and identified a number of nodes equal to $5 \times \sqrt{n}$ where n is the total number of selected events. With 394,833 events, the optimal number of nodes was 2814. The next challenge was defining an appropriate map size for the input data. We selected a map size of 21×134 based on the eigenvalues and eigenvectors [18].

After defining the SOM map, we designed the architecture of the deep auto-encoder to have nine layers. In the training phase, the model was taught to reduce reconstruction errors with a gradient descent optimizer. After training the model, the encoder layer extracted the features and, in so doing, reduced the number of input features.

The second step in a two-stage clustering algorithm is partitive clustering. In this step, we used various partitive clustering algorithms to find the best clustering algorithm with the highest performance. A key concern in developing a partitive clustering algorithm is finding the optimal number of clusters as a suitable number of clusters can improve performance. However, because the number of clusters is generally unknown in real-world problems, we developed Algorithm 4 to address this issue. In brief, the algorithm applies the sum of square error (SSE) and a bootstrapping technique to find a robust result.

C. *Performance Evaluation*

To determine the optimal clustering algorithm for the framework, and for mobile telematics data in general, we compared five different partitive clustering algorithms against three metrics with a five-fold cross-validation method and

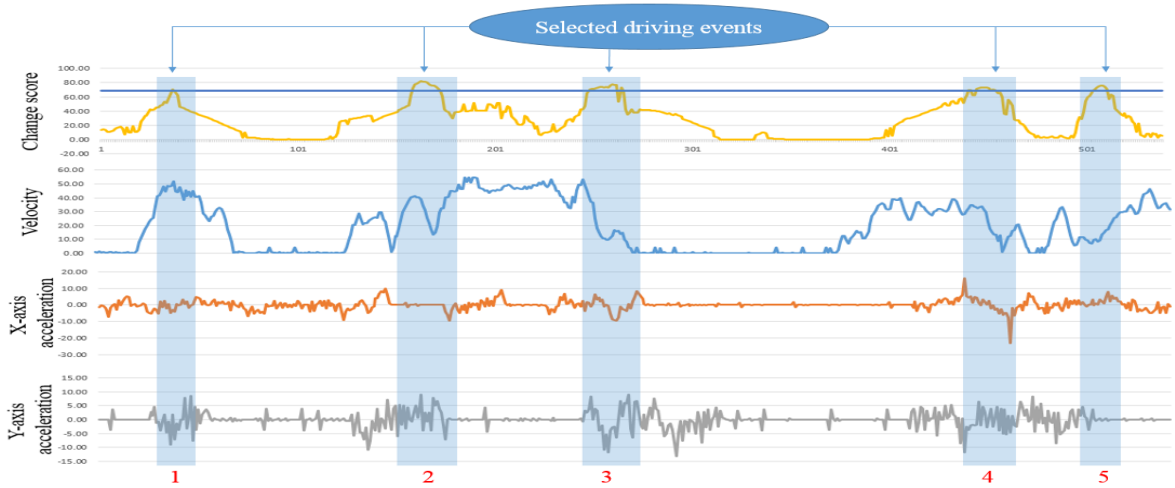

Fig. 5- The change detection scores. The five shaded blue columns indicated a detected driving behavior that exceeded the change threshold (in this case 68.60). All of these events have significant changes in driving behavior over velocity, x-acceleration, and/or y-acceleration. For example, the driver in Event 1 is driving at high and radically varying speeds, with many variations in y-axis acceleration. Event 4 displays high variations in all the variables.

Algorithm 3. The details follow.

The five algorithms we chose for comparison were k-means, MINIbatch k-means, agglomerative clustering, spectral clustering, and BIRCH clustering. After preparing data for clustering using the SOM and DAE, we used the test samples to compare the performance of the five models in terms of execution time, the Calinski Harabasz, and the Davis Boulding indexes.

1) Execution time is the total time to determine the results – smaller values are better.
2) Calinski Harabasz (CH) is a score calculated by assigning N data objects $X = \{x_1, x_2, \dots, x_n\}$ to K different clusters $C = \{c_1, c_2, \dots, c_n\}$ using the following equation [49]:

$$\mathrm{CH(k)} = \frac{Tr(S_B)}{Tr(S_W)} \times \frac{N-k}{k-1}$$
$$Tr(S_B) = \sum_{i=1}^{K} N_i \|m_i - m\|^2 \qquad (12)$$
$$Tr(S_w) = \sum_{i=1}^{K} \sum_{j=1}^{N_i} \|x_j - m_i\|^2$$

where k is the number of clusters, i is the number of items in a cluster n_i, and m_i is the centroid of cluster i. Tr(SB) shows the sum of between-cluster distances, and Tr(Sw) is the sum of within-cluster distances.

3) Davies-Boulding (DB) is another performance index that evaluates the clusters based on the sum of within-cluster scatters and between-cluster separations [50]:

$$\mathrm{DBI} = \frac{1}{n} \sum_{i=1, i \neq j}^{n} max \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \qquad (13)$$

where n is the number of clusters, σ_i is the average distance of all members of the i-th cluster to the center of the j-th cluster, and σ_j is the average distance of all members of cluster j to the center of the i-th cluster. d(ci,cj) is the distance between the center of the two clusters i and j [51, 52].

Algorithm 3: Performance validation algorithm

Input: selected events from change detection
Output: performance results

1. *X = data from change detection*
2. *X_SOM =training SOM model*
3. *X_DEA = training Deep Auto-encoder*
4. *Clustering_methods = [K-means, MinibatchKM, Spectral, Agglomerative, birch]*
5. *For fold in 5-fold cross validation for (X_SOM, X_DEA)*
   - 5.1. *For clustering in Clustering methods*
     - 5.1.1. *Finding optimal number of cluster for clustering by using train data with algorithm 4*
     - 5.1.2. *CH[clustering , fold] = the value of Calinski Harabasz for current fold by using test data*
     - 5.1.3. *DB[clustering , fold] = average value of Davis Boulding for current fold by using test data*
6. *CH[clustering] = average of CH value in all folds for all clutsering_methods*
7. *BD[clustering] = average of BD value in all folds for all clutsering_methods*

Algorithm 4: Finding optimal number of clusters

Input: low complexity data, max number of clusters C, max iterations n
Output: optimal number of clusters

1. *X = data from SOM nodes*
   - 1.1. *For k = 2 to C*
     - 1.1.1. *For i = 1 to n*
     - 1.1.2. *Di = Random under sampling for 80%*
     - 1.1.3. *Clustering Di into k clusters*
     - 1.1.4. *SSEi =Sum of Square errors*
2. $SSE_k = {}^{1}/_{N} \sum SSE_i$
3. *k = the first k which has the amount of improvement rather than previous k is less than 1%*

D. Experimental Results

The results of the comparison follow, starting with the Davis Boulding index in Table II. As shown, SOM did better than the deep auto-encoder with all five clustering algorithms. Within SOM, the performance from best to worst was k-means, Birch, MINIbatchKM, agglomerative, and spectral clustering. Similarly, k-means clustering had an outstanding performance in DAE part in comparison to others. Notably, k-means had the lowest standard deviations and was also the most stable in different folds with both SOM and the deep auto-encoder. Therefore, from a Davis Boulding point of view, SOM + k-means is the optimal choice.

TABLE II- DAVIS BOULDING INDEX RESULTS

Feature extraction	Clustering	Average	Minimum	Maximum	std
SOM	K-means	**0.120**	0.112	**0.130**	0.008
	MINIbatchKM	0.140	0.116	0.177	0.023
	Spectral	0.172	0.125	0.305	0.075
	Agglomerative	0.140	**0.107**	0.162	0.025
	Birch	0.132	0.114	0.155	0.017
Deep Auto-encoder	K-means	0.154	0.148	0.160	**0.005**
	MINIbatchKM	0.165	0.142	0.189	0.018
	Spectral	0.204	0.158	0.317	0.064
	Agglomerative	0.169	0.138	0.199	0.028
	Birch	0.159	0.140	0.184	0.019

The results against the Calinski Harabasz index are shown in Table III. Again, k-means clustering had the highest average CH score with a reasonable standard deviation with both SOM and the deep auto-encoder. In this case, DAE+k-means clustering placed first, followed by SOM+k-means. The SOM+Spectral clustering is the third method with a high CH score, but the standard deviation for this model is very high. From a deeper analysis of each fold, we found this algorithm was always unstable.

TABLE III- - CALINSKI HARABASZ INDEX RESULTS

Feature extraction	Clustering	Average	Minimum	Maximum	std
SOM	K-means	17,375.60	16,754.93	17,891.93	475.62
	MINIbatchKM	15,136.85	14,784.21	15,765.54	381.08
	Spectral	17,040.31	15,191.46	**19,364.84**	1962.88
	Agglomerative	14,871.24	14,714.06	15,010.68	138.09
	Birch	14,753.34	14,528.51	15,171.88	**255.69**
Deep Autoencoder	K-means	**18,242.75**	**17,248.70**	18,967.43	640.11
	MINIbatchKM	16,266.25	15,744.51	16,813.88	480.94
	Spectral	16,292.00	14,336.08	18,900.63	1671.52
	Agglomerative	15,582.44	15,178.67	16,010.59	408.40
	Birch	15,522.14	15,035.29	16,199.02	457.58

Table IV shows the execution times. Efficiency is an important factor since data on driving characteristics tends to be very large-scale, and running unsupervised learning algorithms are prone to long runtimes.

TABLE IV- EXECUTION TIME RESULTS (MINUTES)

	SOM	Deep Auto-encoder
K-means	40.87	180.93
MINIbatchKM	**33.47**	**66.99**
Spectral	82.35	250.70
Agglomerative	41.03	199.38
Birch	41.30	190.73

As the results show, SOM had much faster running times with all clustering methods than the deep auto-encoder. The most important reason for this difference is that, with a deep auto-encoder, one record equals one point while, with SOM, one point equals an abstracted group of records.

Across all three metrics, the optimal clustering choice for driving style pattern recognition is clear – SOM + k-means, firstly because it had a very low DB index in comparison to other methods, which means that the extracted clusters with SOM+k-means are unique and they are less similar to other clusters in comparison to other techniques [53]. In addition, CH index in deep auto encoder is slightly better than SOM+k-means, and this difference is not big enough to encourage us for select this algorithm as the selected method while its computation cost is very high and BD index is very low.

VI. EXTRACTED DRIVING PATTERNS

From the three tests in the previous section, we determined that k-means in tandem with SOM was the best overall algorithm for recognizing driving style patterns. The next step is finding the optimal number of clusters. We used Algorithm 4 to determine the optimal number of clusters by using SOM+k-means clustering algorithm. We found that the optimal number of clusters was 29. Fig. 5 lists the sum of square errors for different numbers of clusters in each iteration, showing 29 as the optimum number of clusters, because 29 clusters does not exceed the defined threshold of 1%, and it does meet the stopping condition of less than 1% improvement.

Hence, we extracted 29 unique driving behaviors from our data set. Each cluster is a group of time-series data and raw numbers. As results, however, raw numbers do not mean much to transportation experts so we need to understand each driving pattern and find a meaningful name for each cluster. We followed a matching algorithm to find a suitable label for each driving behavior. In this algorithm, first, we reviewed various driving behavior introduced by top-ranked, highly cited publications and selected three papers to review: [1-3]. Second, we developed a descriptive analytics to understand each category using average, minimum, maximum, and standard deviation across velocity, acceleration, x-axis and y-axis acceleration. Then, we compared the similarity between the extracted driving patterns and current driving behaviors in the literature. We then selected the most similar pattern in the literature as a representative of each category. Finally, we named each cluster to reflect the name of the most similar driving behavior found in the literature.

Algorithm 5: The matching algorithm

Input: Extracted driving behavior, current driving behavior in the literature

Output: Corresponding driving pattern in the literature, and Name of all clusters

1. *DB = 29 extracted driving behaviors by SOM + k-means*
2. *DB_lit = various driving behavior in the current literature*
3. *For each driving_behavior in DB:*
   - 3.1. *Developing descriptive analytics with minimum, maximum, average, and standard deviation across all input variables*
   - 3.2. *Corresponding_patterns[driving_behavior] = the most similar pattern in the literature for driving_behavior*
   - 3.3. *Names[driving_behavior] = finding a suitable name according to the corresponding driving behavior*
     - 3.3.1. *DB[clustering , fold] = average value of Davis Boulding for current fold by using test data*
4. *Return Corresponding_patterns, names*

After implementing algorithm 5, we understand the characteristics of all 29 clusters. For example, Cluster 17 represents those who drive at very low speed with low acceleration, and accounts for 16.5% of the events. This behavior shows that these drivers have a tendency to stop the vehicle. In another driving group, Cluster 29, the y-acceleration

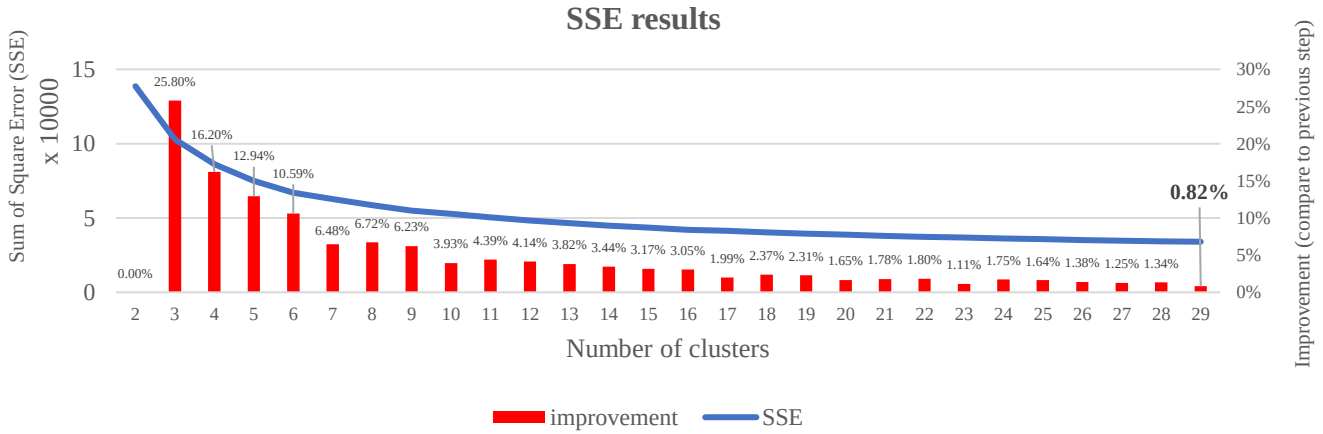

Fig. 6- Sum of Square error per number of cluster for SOM + k-means clustering

is close to zero and the x-acceleration is higher than zero for a short period of time, which is similar to the cornering behavior described by Fazeen et al. [2]. The next group, Cluster 13, is normal driving behavior, i.e., standard speeds with very low acceleration, few changes, and a small standard deviation. Yu et al. [3] describe this type of driving as "normal driving behavior". Cluster 8 exhibits swerving behavior as described by Chen et al. [1]. The value of both x-axis and y-axis acceleration have a high peak value with a high standard deviation. Drivers in Cluster 2 exhibit weaving behavior at high speeds. They have a very high variation between x- and y-axis acceleration. The standard deviation of acceleration is very high, and the mean value of acceleration is high [3]. Cluster 6 reflects sudden breaking and accounts for 4.3% of the driving events. x-acceleration remains unchanged while y- acceleration significantly decreases, and the standard deviation of the y-acceleration is high [1]. Cluster 26 is characterized by high variations in both x- and y-acceleration. The velocity range is medium, and the standard deviation of acceleration is high with low acceleration. In the above, we explained the top clusters that account for the 50% of driving events to describe how the clustering results are matched with a corresponding name in the transportation research. Additional information about all the other clusters, along with the statistical results for each group, are provided in the Appendix.

VII. Conclusion and Further Study

Understanding driving patterns with unsupervised learning techniques is an underexplored area of research and finding the best clustering algorithm with the highest performance and the optimal number of clusters is still problematic. In this paper, we proposed a framework to extract unique driving behaviors from smartphones generated data. Previous research on driving style analytics is largely based on laboratory simulations, site investigations, or questionnaires. These datasets are completely different from real-world data, which are mostly unlabeled. Further, when real data is used, it comes from in-vehicle recorders, which are expensive to install and limits the take-up of telematics. Our framework, however, was developed using real smartphone data rather than in-vehicle data recorders or a synthetic dataset, which is less expensive.

In our framework, we used the relative unconstrained least-squares importance fitting (RuLSIF) model as a change detection algorithm to detect the most informative time frames for recognizing driving characteristics. We used this algorithm to decrease the complexity of clustering algorithms by removing unnecessary time frames. A two-stage clustering framework comprising SOM and a deep auto-encoder to reduce the complexity of input data, after which the data can be clustered for analysis with a partitive clustering algorithm. From an evaluation of k-means, MINIbatch k-means, agglomerative clustering, spectral clustering, and Birch clustering, we find that SOM + k-means clustering is the optimal choice for extracting driving patterns according to the result of the Davis Boulding and Calinski Harabasz indexes and execution time. The final clustering results revealed 29 unique driving categories. We found the most similar driving patterns in the literature for each cluster to identify a label for all the extracted driving categories.

In the future, we plan to propose an unsupervised decision support system that uses extracted driving categories as the criteria for automatic decision making. To ensure the system is comprehensive and effective, we will need to design a risk assessment framework that can evaluate the probability and severity of each pattern and calculate a risk score for each unique behavior and, in turn, each driver. Fuzzy inference systems will be our starting point in this endeavor. Moreover, one of the most vexing challenges with using machine learning techniques for driving style analytics is the lack of labeled data. Thus, researchers in the field of transportation and road safety could also use this framework to label unlabeled driving patterns in telematics data. Once labelled, the data could be used with a supervised learning technique with the most state of art machine learning algorithms for various applications.

Appendix

Table V
Frequent Driving Behaviors - part I

Cluster number	Frequency Percentage (%)	Average speed (km/h)	Average acceleration (m/s^2)	Acceleration std	Name	Patterns of behavior
17	16.49%	1.961	-0.108	0.088	Warm stopping	Drivers in this group drive at low speeds with low deceleration and are prone to stopping.
29	9.08%	45.808	0.353	0.202	Cornering with medium speed	During cornering behavior, y- acceleration is close to zero. x-acceleration is high with significant standard deviation [2].
13	6.68%	50.550	0.333	0.022	Driving at normal speed	Drivers proceed at normal speed with very low acceleration and standard deviation.
8	5.18%	39.130	0.011	1.288	Swerving at medium speed	While swerving, x- and y-acceleration both have a high peak and high standard deviation [1].
2	4.96%	80.925	0.002	2.225	Weaving at high speed	There is high variation in between x- and y-acceleration. y-acceleration is very smooth over an extended period. The standard deviation of acceleration is high, but with a low mean [3].
11	4.59%	87.621	0.1114	0.163	Cornering at high speed	During cornering, speeds are high and x-acceleration increases rapidly over a short period of time, while y- acceleration is almost zero [3].

TABLE VI
FREQUENT DRIVING BEHAVIORS - PART II

Cluster number	Frequency Percentage (%)	Average speed (km/h)	Average acceleration (m/s^2)	Acceleration std	Name	Patterns of behavior
27	4.54%	70.784	-0.625	0.1769	Sideslipping with high speed	y-axis acceleration fell down sharply, the minimum and average value of y-axis acceleration is negative and x-axis acceleration is not near to zero[1].
23	4.49%	76.21	0.0315	2.2296	weaving with high speed	There is high variation in between x- and y-acceleration. y-acceleration is very smooth over an extended period. The standard deviation of acceleration is high, but with a low mean [3].
6	4.37%	61.207	-2.258	1.257	Sudden braking from a high speed	During sudden braking, x-acceleration remains unchanged, and y- acceleration decreases significantly. Standard deviation in y-acceleration is very high [1].
26	3.86%	56.389	0.190	2.039	Weaving at medium speed	There is a high variation between x- and y-acceleration. y-acceleration is very smooth over extended periods. The standard deviation of acceleration is high. The mean value of acceleration is very low [3].
12	3.39%	65.784	-0.381	0.151	Cornering at high speed	During cornering, speeds are high and x-acceleration increases rapidly over a short period of time, while y- acceleration is almost zero [3].
18	3.10%	40.558	6.838	0.963	High acceleration behavior	Acceleration is high over a very short time reflective of sudden maneuvering habits [2].
16	2.87%	95.259	0.242	0.123	Cornering at very high speed	During cornering, speeds are very high and x-acceleration increases rapidly over a short period of time, while y- acceleration is almost zero [3].
3	2.43%	8.499	-1.157	1.179	Cornering at low speed	During cornering, speed is low and x-acceleration increases significantly for a short period of time. y- acceleration is almost zero [3].
28	2.34%	26.679	0.27 8	1.307	Changing lanes at low speed	Y-acceleration decreases before changing lanes and increases afterwards [2].
22	2.20%	32.458	-1.578	0.797	Sudden braking from a medium velocity	During sudden braking, x-acceleration remains unchanged, and y- acceleration decreases significantly. Standard deviation in y-acceleration is very high [3].
14	1.95%	12.707	-6.173	1.730	Sideslipping with low speed	y-acceleration decreases sharply with a high range in x-acceleration and negative value. All these behaviors occur within a very short period [1].
9	1.88%	12.576	-7.108	1.836	Sudden braking from a low velociy	During sudden braking, the x-acceleration remains unchanged, and the y-acceleration decreases significantly. The standard deviation of the y-acceleration is very high [3].
10	1.82%	19.547	0.575	1.030	Left lane change	A left lane change is formed by a small decrease in the value of x-axis acceleration, followed by an increase in x-axis acceleration [2].
25	1.81%	40.335	6.273	2.628	Sudden starting	Very high acceleration starting from a stop up to a very high average speed.
15	1.80%	56.533	-7.213	1.447	Sideslipping with medium speed	y-acceleration decreases sharply with a high range in x-acceleration and negative value. All these behaviors occur within a very short period [1].
7	1.71%	34.481	1.096	1.616	Right lane change	A right lane change is formed by a small increase in the value of x-axis acceleration, followed by a decrease in x-axis acceleration [2].
1	1.63%	37.197	-2.111	0.842	Safe cornering	Drivers drive at medium speed and decrease their speed before turning.
5	1.50%	107.15	5.160	0.154	Turning with a wide radius	High and rapid x-acceleration with a y-acceleration of close to zero. The x-acceleration mean is far from zero [3]
19	1.37%	32.098	-9.408	2.055	Unknown	Driving pattern in this group is unknown and is not understandable for experts.
24	1.29%	44.609	-10.51	2.087	Fast U-turn	A rapid rise in x-acceleration to a very high value followed by a rapid drop to a very low value for a short period of time. The standard deviation acceleration is high [1].
4	0.98%	52.887	-6.621	1.975	Sudden braking	Speed decreases over a very short time. This type of sudden braking is much more dangerous than in Cluster 6 because the deceleration is much higher.
20	0.96%	29.389	-1.064	1.131	Turn right cornering	Drivers decrease their speed and acceleration changes from the x-axis to the y-axis. This group could be merged with cluster number 26.
21	0.73%	120.49	-0.106	0.152	Very high speed with low decleration	Driving at very high speed without any significant changes.

This table shows the 29 clusters extracted with SOM+ K-means, which represent the driving patterns within our dataset. The average speeds and acceleration values are the means of the velocity and instantaneous acceleration figures for all driving patterns in each group. The names for each group were derived from the patterns and values at each cluster center with reference to previous studies in transportation.

REFERENCES

[1] Z. Chen, J. Yu, Y. Zhu, Y. Chen, and M. Li, "D 3: abnormal driving behaviors detection and identification using smartphone sensors," in 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2015, pp. 524-532.

[2] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M. C. González, "Safe driving using mobile phones," IEEE Transactions on Intelligent Transportation Systems, vol. 13, pp. 1462-1468, 2012.

[3] J. Yu, Z. Chen, Y. Zhu, Y. J. Chen, L. Kong, and M. Li, "Fine-grained abnormal driving behaviors detection and identification with

smartphones," IEEE Transactions on Mobile Computing, vol. 16, pp. 2198-2212, 2017.
[4] J. Wahlström, I. Skog, and P. Händel, "Smartphone-based vehicle telematics: A ten-year anniversary," IEEE Transactions on Intelligent Transportation Systems, vol. 18, pp. 2802-2825, 2017.
[5] P. G. Malalur, H. Balakrishnan, and S. R. Madden, "Telematics using personal mobile devices," ed: Google Patents, 2013.
[6] J. Wahlström, I. Skog, P. Händel, B. Bradley, S. Madden, and H. Balakrishnan, "Smartphone Placement Within Vehicles," IEEE Transactions on Intelligent Transportation Systems, 2019.
[7] P. Desyllas and M. Sako, "Profiting from business model innovation: Evidence from Pay-As-You-Drive auto insurance," Research Policy, vol. 42, pp. 101-116, 2013.
[8] Y. Zhao, "Telematics: safe and fun driving," IEEE Intelligent systems, vol. 17, pp. 10-14, 2002.
[9] Y. Zhao, "Mobile phone location determination and its impact on intelligent transportation systems," IEEE Transactions on intelligent transportation systems, vol. 1, pp. 55-64, 2000.
[10] B. F. Bowne, N. R. Baker, D. L. Marzinzik, M. E. Riley, N. U. Christopulos, B. M. Fields, et al., "Methods to Determine a Vehicle Insurance Premium Based on Vehicle Operation Data Collected Via a Mobile Device," ed: Google Patents, 2013.
[11] Y. Song, J. Lu, H. Lu, and G. Zhang, "Fuzzy clustering-based adaptive regression for drifting data streams," Accepted by IEEE Transactions on Fuzzy Systems, 2019.
[12] H. Guo, Y. Ji, T. Qu, and H. Chen, "Understanding and modeling the human driver behavior based on MPC," IFAC Proceedings Volumes, vol. 46, pp. 133-138, 2013.
[13] J. Lee and K. Jang, "A framework for evaluating aggressive driving behaviors based on in-vehicle driving records," Transportation Research Part F: Traffic Psychology and Behaviour, 2017.
[14] M. Basseville and I. V. Nikiforov, Detection of abrupt changes: theory and application vol. 104: Prentice Hall Englewood Cliffs, 1993.
[15] N. Itoh and J. Kurths, "Change-point detection of climate time series by nonparametric method," in Proceedings of the world congress on engineering and computer science, 2010, pp. 20-23.
[16] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," Neural Networks, vol. 43, pp. 72-83, 2013.
[17] T. Kohonen, "The self-organizing map," Proceedings of the IEEE, vol. 78, pp. 1464-1480, 1990.
[18] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," IEEE Transactions on neural networks, vol. 11, pp. 586-600, 2000.
[19] H.-k. Du, J.-x. Cao, Y.-j. Xue, and X.-j. Wang, "Seismic facies analysis based on self-organizing map and empirical mode decomposition," Journal of Applied Geophysics, vol. 112, pp. 52-61, 2015.
[20] P. Mangiameli, S. K. Chen, and D. West, "A comparison of SOM neural network and hierarchical clustering methods," European Journal of Operational Research, vol. 93, pp. 402-417, 1996.
[21] H. Zhang, T. W. Chow, and Q. J. Wu, "Organizing books and authors by multilayer SOM," IEEE transactions on neural networks and learning systems, vol. 27, pp. 2537-2550, 2016.
[22] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," Information Fusion, vol. 42, pp. 146-157, 2018.
[23] Y. Xiao and J. Yu, "Partitive clustering (K-means family)," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 2, pp. 209-225, 2012.
[24] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, et al., "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," IEEE transactions on emerging topics in computing, vol. 2, pp. 267-279, 2014.
[25] A. Namvar, M. Ghazanfari, and M. Naderpour, "A customer segmentation framework for targeted marketing in telecommunication," in Intelligent Systems and Knowledge Engineering (ISKE), 2017 12th International Conference on, 2017, pp. 1-6.
[26] N. Lu, H. Lin, J. Lu, and G. Zhang, "A customer churn prediction model in telecom industry using boosting," IEEE Transactions on Industrial Informatics, vol. 10, pp. 1659-1665, 2014.
[27] K. Wang, X. Qi, H. Liu, and J. Song, "Deep belief network based k-means cluster approach for short-term wind power forecasting," Energy, 2018.
[28] S. Khanmohammadi, N. Adibeig, and S. Shanehbandy, "An improved overlapping k-means clustering method for medical applications," Expert Systems with Applications, vol. 67, pp. 12-18, 2017.
[29] G. Vaia, E. Carmel, W. DeLone, H. Trautsch, and F. Menichetti, "Vehicle Telematics at an Italian Insurer: New Auto Insurance Products and a New Industry Ecosystem," MIS Quarterly Executive, vol. 11, 2012.
[30] P. Handel, I. Skog, J. Wahlstrom, F. Bonawiede, R. Welch, J. Ohlsson, et al., "Insurance telematics: Opportunities and challenges with the smartphone solution," IEEE Intelligent Transportation Systems Magazine, vol. 6, pp. 57-70, 2014.
[31] Z. Shou and X. Di, "Similarity analysis of frequent sequential activity pattern mining," Transportation Research Part C: Emerging Technologies, vol. 96, pp. 122-143, 2018.
[32] C. Saiprasert, T. Pholprasit, and S. Thajchayapong, "Detection of driving events using sensory data on smartphone," International journal of intelligent transportation systems research, vol. 15, pp. 17-28, 2017.
[33] W. Wang and J. Xi, "A rapid pattern-recognition method for driving styles using clustering-based support vector machines," in American Control Conference (ACC), 2016, 2016, pp. 5270-5275.
[34] M. Henriksson, "Driving context classification using pattern recognition."
[35] W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang, "Characterizing Driving Styles with Deep Learning," arXiv preprint arXiv:1607.03611, 2016.
[36] W. Dong, T. Yuan, K. Yang, C. Li, and S. Zhang, "Autoencoder regularized network for driving style representation learning," in Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 1603-1609.
[37] M. Siami, M. Naderpour, and J. Lu, "A Choquet Fuzzy Integral Vertical Bagging Classifier for Mobile Telematics Data Analysis," in 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2019, pp. 1-6.
[38] L. Moreira-Matias and H. Farah, "On developing a driver identification methodology using in-vehicle data recorders," IEEE Transactions on Intelligent Transportation Systems, vol. 18, pp. 2387-2396, 2017.
[39] H. Liu, T. Taniguchi, Y. Tanaka, K. Takenaka, and T. Bando, "Visualization of driving behavior based on hidden feature extraction by using deep learning," IEEE Transactions on Intelligent Transportation Systems, 2017.
[40] O. Shouno, "Deep unsupervised learning of a topological map of vehicle maneuvers for characterizing driving styles," in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2917-2922.
[41] T. T. Nguyen, P. Krishnakumari, S. C. Calvert, H. L. Vu, and H. van Lint, "Feature extraction and clustering analysis of highway congestion," Transportation Research Part C: Emerging Technologies, vol. 100, pp. 238-258, 2019.
[42] Z. Zhou, W. Dou, G. Jia, C. Hu, X. Xu, X. Wu, et al., "A method for real-time trajectory monitoring to improve taxi service using GPS big data," Information & Management, vol. 53, pp. 964-977, 2016.
[43] G. L. Foresti, M. Farinosi, and M. Vernier, "Situational awareness in smart environments: socio-mobile and sensor data fusion for emergency response to disasters," Journal of Ambient Intelligence and Humanized Computing, vol. 6, pp. 239-257, 2015.
[44] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 5, pp. 114-127, 2012.
[45] S. Aminikhanghahi, T. Wang, and D. J. Cook, "Real-time change point detection with application to smart home time series data," IEEE Transactions on Knowledge and Data Engineering, 2018.
[46] M. Saraee, S. Vahid Moosavi, and S. Rezapour, "Application of Self Organizing Map (SOM) to model a machining process," Journal of Manufacturing Technology Management, vol. 22, pp. 818-830, 2011.
[47] X. Zhang, X. Zhao, and J. Rong, "A study of individual characteristics of driving behavior based on hidden markov model," Sensors & Transducers, vol. 167, p. 194, 2014.
[48] R. Céréghino and Y.-S. Park, "Review of the self-organizing map (SOM) approach in water resources: commentary," Environmental Modelling & Software, vol. 24, pp. 945-947, 2009.
[49] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," Communications in Statistics-theory and Methods, vol. 3, pp. 1-27, 1974.
[50] D. L. Davies and D. W. Bouldin, "A cluster separation measure," IEEE transactions on pattern analysis and machine intelligence, pp. 224-227, 1979.

[51] Z. Halim, M. Waqas, A. R. Baig, and A. Rashid, "Efficient clustering of large uncertain graphs using neighborhood information," International Journal of Approximate Reasoning, vol. 90, pp. 274-291, 2017.
[52] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, pp. 1650-1654, 2002.
[53] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in 2010 IEEE International Conference on Data Mining, 2010, pp. 911-916.

Mohammad Siami is currently pursuing a PhD with the Decision Systems and e-Service Intelligent (DeSI) Research Laboratory, Center for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. He has ten years of experience in solving risk assessment with artificial intelligence and machine learning techniques in financial service companies including banking and insurance. His current research interests include artificial intelligence, machine learning, and smartphone data analytics.

Mohsen Naderpour (M'16) received his PhD from the University of Technology Sydney and currently is a Lecturer at the School of Information, Systems and Modelling. He is also a core member of the Centre for Artificial Intelligence and the Center for Advanced Modelling and Geospatial Information Systems. Mohsen began his professional life as a safety professional in high risk industries including transportation and oil before taking up a position in academia as a research fellow with the Global Big Data Technologies Centre at UTS. His research areas include decision support systems, risk analysis, uncertain information processing, and data analytics.

Jie Lu (F'18) received her PhD in information systems from Curtin University, Australia, in 2000. She is currently a Distinguished Professor, Director of the Center for Artificial Intelligence, and the Associate Dean (Research Excellence) of the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. She has published 10 research books and over 400 papers in refereed journals and conference proceedings, with over 170 papers in IEEE Transactions and other international journals. She has received over 20 Australian Research Council Discovery Project grants and many other research projects. Her current research interests include fuzzy transfer learning, decision support systems, recommender systems, and concept drift. Prof. Lu is a fellow of the International Journal of Fuzzy Systems. She has served as a guest editor of 12 special issues and general/PC/organization chairs for 10 international conferences and delivered 20 keynote/plenary speeches at the IEEE and other international conferences. She serves as an Editor-in Chief for *Knowledge-Based Systems* (Elsevier, the *International Journal on Computational Intelligence Systems* (Atlantis) and is an Associate Editor for *IEEE Transactions on Fuzzy Systems*.