This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

Clustering Driving Encounter Scenarios Using Connected Vehicle Trajectories

Wenshuo Wang, *Member, IEEE,* Aditya Ramesh, and Ding Zhao

***Abstract*—Classification and analysis of driving behaviors offer in-depth knowledge to make an efficient decision for autonomous vehicles. This paper aims to cluster a wide range of driving encounter scenarios based only on multi-vehicle GPS trajectories. Towards this end, we propose a generic unsupervised learning framework comprising two layers: feature representation layer and clustering layer. In the layer of feature representation, we combine the deep autoencoders with a distance-based measure to map the sequential observations of driving encounters into a computationally tractable space, which quantifies the spatiotemporal interaction characteristics of two vehicles. The clustering algorithm is then applied to the extracted representations to cluster homogeneous driving encounters into groups. Our proposed generic framework is then evaluated using 2,568 naturalistic driving encounters. Experimental results show that our proposed generic framework incorporated with unsupervised learning can cluster multi-trajectory data into distinct groups. These clustering results could benefit decision-making and design of autonomous vehicles.**

***Index Terms*—Multi-vehicle behavior clustering, autoencoder, vehicle trajectory, unsupervised learning.**

I. Introduction

MULTI-VEHICLE interaction is an everyday and important driving scenario in real traffic, but challenging to be analyzed and modeling because of the diversity in spatiotemporal attributes. Driving encounter in this paper is referred to as the scenario where two or multiple vehicles are spatially close to and interact with each other. In real life, autonomous vehicles should make proper decisions in various negotiation scenarios such as on highways and at intersections. An expected decision-making and control system for fully autonomous driving can adequately understand and handle all possible driving encounters to guarantee road safety and traffic efficiency. However, the diversity of encountering scenarios in real life is overwhelming, as illustrated in Fig. 1.

Multi-vehicle interaction modeling and analysis rely on many kinds of advanced communication techniques such as dedicated short-range communications (DSRC) for driving encounter data collection. The multi-vehicle interaction can be interpreted by their trajectories – a set of positional information for moving vehicles, ordered by time. The growing use of global positioning system (GPS) receivers in equipped vehicles empowers researchers to collect large amounts of high-quality trajectory data at a low cost. Using positional trajectories with GPS data allows us to dig underlying information and visualize multi-vehicle behaviors with road context. Vehicle trajectories represented by GPS data have also been widely used for driver behavior pattern analysis and prediction [1]–[4], travel destination prediction [5], anomalous driving behavior detection [6], eco-driving [7], vehicle behavior reconstruction [8]. However, the diversity of driving encounters at various driving conditions (e.g., road, traffic, and other road users) and the flood of high-dimensional traffic data overwhelms human insight and analysis [9], thereby limiting the full use of these data. Fig. 1 presents some typical driving encounters consisting of two vehicles at different intersections. Typically, clustering driving encounters into distinct groups helps us to identify the common underlying points, encourages a vibrant design of space and functionality for synthetic traffic systems, and highlights a compelling need for the integrative analysis of interactions between human drivers or between the human-driven vehicles and autonomous vehicles.

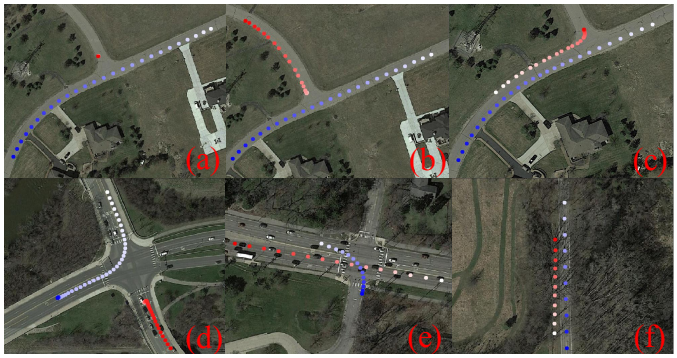


Fig. 1. Examples of driving encounters at (a)-(c) T-shape intersections, (d)-(e) cross-intersections, and (f) a straight road. Thick red and blue dots are the start position of vehicles, and white dots are the end position of vehicles.

Many clustering algorithms and data mining techniques have been implemented to analyze vehicle trajectories and associated applications, as reviewed in [10], [11]; however, most of them aim to cluster single-vehicle trajectories, rather than multi-vehicle trajectories. For example, Besse *et al.* [12] proposed a distance-based algorithm to learn representations of the single vehicle trajectory and then applied hierarchical clustering algorithms. The authors in [5] developed a two-step procedure to predict a trip's destination using a density-based

The corresponding author is Ding Zhao.

W. Wang is with the Department of Mechanical Engeering, Carnegie Mellon University (CMU), Pittsburgh, PA. He was with the Department of Mechanical Engineering, University of Michigan (UM), Ann Arbor, MI, 48109. USA. e-mail: wwsbit@gmail.com.

A. Ramesh is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109 USA e-mail: raaditya@umich.edu.

D. Zhao is with the Department of Mechanical Engeering and also the Robotics Institute, Carnegie Mellon University (CMU), Pittsburgh, PA, USA: dingzhao@cmu.edu.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

methods using the start and end portions of trajectories. Yao *et al*. [13] used a sliding window to extract a set of moving behavior features that capture space and time-invariant characteristics of the trajectories. Zhao *et al*. [14], [15] extracted lane change behavior according to the vehicle's raw movement trajectory and conducted the scene-based analysis. In order to obtain traffic patterns for traffic surveillance, Choong *et al*. [16] implemented the longest common subsequence (LCSS) to measure similarity levels of trajectories as features and then clustered vehicle trajectories into groups using k-means and fuzzy c-means. Atev, *et al*. [17] developed a method suitable for clustering of single-vehicle trajectories, which reveals the semantic structure of the scene (e.g., road structures [18]) and allows inference about the interactions between road users, thus providing an accurate and adequate representation of the environment for making decisions [19]. Some studies [11], [20]–[24] also applied clustering algorithms to detect traffic information such as traffic hotspots, traffic patterns, mutli-vehicle interactive patterns and traffic volume based on vehicle GPS trajectories.

However, all these methods above are unsuitable for clustering a set of multi-vehicle trajectories because of their limited ability to represent high-dimensional sequential observations. Investigating multi-vehicle interaction patterns brings insights into how human drivers make decisions when negotiating with others, thereby lays foundation for self-driving. For instance, classifying and analyzing negotiations behaviors at unsignalized intersections would offer researchers detailed and applicable information to design rule-based decision-making systems that harmoniously negotiate with surrounding road users [25], [26]. Towards this goal, many researchers used vehicle-to-vehicle (V2V) trajectories in the empirically predefined specific scenarios such as changing lanes. Although achieved remarkable progress has been made, the diversity of scenarios concerned are limited. Therefore, categorizing similar driving encounters brings insight into each kind of encounter scenario and offer opportunities to test the existing algorithms. For example, Pokorny *et al*. [27] proposed a topological trajectory clustering by considering the relative persistent homology between motion trajectories. Our previous work [28] directly applied a common autoencoder to extract features of characterizing driving encounters for clustering, but the representations learned were not interpretable. To the best of our knowledge, great efforts have been made to tackle a bunch of single-vehicle trajectories, but few efficient approaches were proposed to cluster a set of multi-vehicle trajectories.

In this paper, we primarily focus on the driving encounter scenario within two vehicles engaged and clustering their GPS trajectories into distinct groups. The main contributions of this paper are threefold.

- Proposed a generic two-layer framework to cluster the driving encounter scenarios represented by multi-vehicle GPS trajectories.
- Developed different unsupervised learning methods to encode driving encounters into computationally tractable measure space using deep autoencoders, distance-based measure, and their combinations.

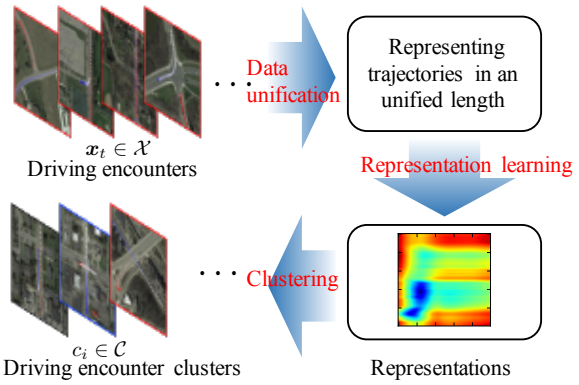


Fig. 2. Our proposed framework of clustering driving encounters.

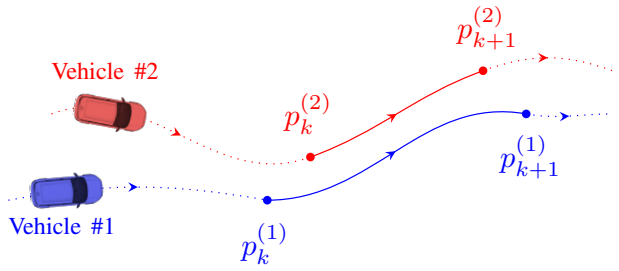


Fig. 3. Illustration of one driving encounter with two vehicle trajectories.

- Validated the proposed two-layer framework using different algorithms based on 2,568 naturalistic driving encounters.

The remainder of this paper is organized as follows. Section II presents the framework to cluster driving encounters. Section III details the approaches to extracting representatives. Section IV presents the clustering method and performance metrics. Section V introduces the experimental procedure and data collection. Section VI analyzes and discusses the experimental results, followed by a conclusion and future work in Section VII.

II. Clustering Framework

In this section, we shall discuss the generic framework of clustering driving encounters, as shown in Fig. 2. This framework can be realized by three steps: driving encounter unification, representation extraction, and clustering. Before detailing the framework, we first introduce the mathematical formulation of driving encounters.

A. Driving Encounter

Two vehicle trajectories with the same duration form a driving encounter, which was usually represented by vectors with a fixed sampling rate.A single driving encounter is represented as

$$\boldsymbol{x} = \left\{(p_1^{(1)}, p_1^{(2)}, t_1), \cdots, (p_k^{(1)}, p_k^{(2)}, t_k), \cdots, (p_K^{(1)}, p_K^{(2)}, t_K)\right\} \tag{1}$$

where $p_k^{(1)} \in \mathbb{R}^2$ and $p_k^{(2)} \in \mathbb{R}^2$ are the positions of two vehicles in a driving encounter at discrete time t_k (Fig. 3), $k = 1, 2, \ldots, K$ is the index of the sampled data points. $\boldsymbol{x}$. In particular, we define $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ as a set of driving encounters with N samples.

B. Driving Encounter Unification

It is convenient to apply clustering on elements with the same size. For real world driving encounters, their lengths could be significantly different from each other. In order to make clustering algorithms practically tractable, we need to scale driving encounters into an identical length with little loss of information. Many ripe approaches [29] to unifying trajectories have been developed. For example:

- Trajectory transformation algorithms, which project the trajectories into a structured space with fixed parameters, for example, linear transformation, curving fitting (e.g., uniform cubic B-spline curve), discrete Fourier transformation, etc.
- Re-sampling methods, which generate trajectory points by re-sampling.
- Trajectory substitute, which utilizes the essential components of trajectories (termed as sub-trajectories) to represent hidden information of original trajectory data.
- Points of interest, which is flexible and preferred when research focuses on some specific regions of surveillance (termed as points of interest) rather than points outside the regions.
- Scale-invariant features, which have been widely used to extract more robust and representative features (e.g., histograms of oriented gradients and histograms of optical flow) from image frames rather than the positional trajectories.

Though the last three approaches could perform very well for a single-vehicle trajectory, it is nontrivial to directly apply them to the pair of multi-vehicle trajectories in driving encounters. Hence, we prefer to use the re-sampling method with interpolation processes as it is very flexible to operate. More calculation details see [29].

C. Representation Learning

Straightforwardly implementing the common clustering algorithms to the sequential observations of driving encounters is practically intractable because it is meaningless to compute the center of multi-vehicle trajectories. In order to solve this problem, we transfer the multi-vehicle trajectories into a new space and then select associated representations to characterize the interaction of vehicles in driving encounters. Manipulating on representations makes it calculable to maximize the similarity of samples within and between clusters. Many different approaches have been developed to learn representations of capturing the temporal and spatial relations of multi-vehicle trajectories, for example, by measuring the distance between two trajectories [5], [12] or by using neural networks to learn representations of individual trajectories [13], [30]. We shall introduce three kinds of approaches to learn representations in Section III.

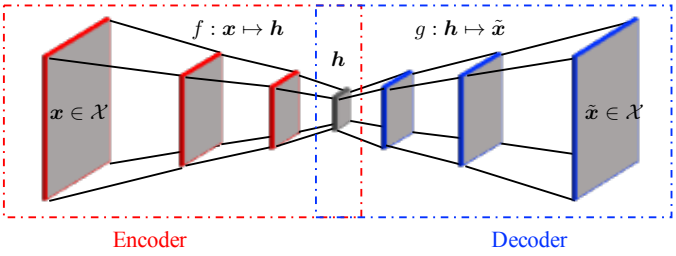


Fig. 4. Illustration of representation learning through the deep neural network-based autoencoder, with an encoder f and a decoder g to learn hidden representations $\boldsymbol{h}$ given observations $\boldsymbol{x} \in \mathcal{X}$.

D. Clustering

These learned representations enable us to make full use of the off-the-shelf clustering algorithms (e.g., k-means) to gather driving encounters into groups. The clustering performance can also be evaluated based on these representations. More details are shown in Section IV.

III. Representation Learning

Clustering similar driving encounters into groups requires to extract the features capable of capturing their primary characteristics. Unlike extracting features of images with specific and explicit labels, we have limited prior knowledge about the feature space of driving encounters. Fortunately, many approaches to learning representations of time-series trajectories have been developed to study sequential data. In this paper, we mainly focus on the types of multi-vehicle dynamic interactions in spatial and temporal spaces. We shall introduce three ways to achieve this: deep autoencoders, distance-based measure, and shape-based measure. The limitations and advantages of these methods are then discussed.

A. Deep Autoencoders

Our goal is to characterize a given driving encounter by finding a representation from which the driving encounter can be completely reconstructed. The autoencoder, which could generate a sample from the learned representation as close as possible to its original input, has been widely used to dig underlying information of time-series data. An autoencoder is a specific neural network (NN) consisting of encoder and decoder (as shown in Fig. 4) that attempts to copy its input $\boldsymbol{x}$ to its output $\tilde{\boldsymbol{x}}$, where the hidden layer $\boldsymbol{h}$ describes the code to represent the input $\boldsymbol{x}$. By training a model to minimize the difference between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, the hidden layer $\boldsymbol{h}$ can capture the underlying characteristics of this driving encounter. Autoencoders vary in the architecture of the encoder and decoder as a consequence of the kind of input data being supplied. For example, a structure of long-short term memory (LSTM) generates an LSTM autoencoder, and a structure of convolutional NN (CNN) creates a convolutional autoencoder. In general, the encoder and the decoder can be formulated as follows.

- *Encoder*: The encoder (red dash box in Fig. 4) is a multilayer neural networks of mapping an input $\boldsymbol{x}_{(i)}$ at the i^{th}-layer into $\boldsymbol{x}_{(i+1)}$ at the $(i+1)^{\text{th}}$-layer by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

$$\boldsymbol{x}_{(i+1)} = \boldsymbol{w}_{(i)}^{\top}\boldsymbol{x}_{(i)} + \boldsymbol{b}_{(i)} \tag{2}$$

where $\boldsymbol{w}_{(i)}$ is the weight matrix for transferring data from the i^{th}-layer to $(i+1)^{\text{th}}$-layer, $\boldsymbol{b}_{(i)}$ is the bias vector. The middle layer of the autoencoder in Fig. 4 is the representation $\boldsymbol{h}$ of driving encounters. Here, the Equation (2) represents a generic form of autoencoders. For specific forms such as LSTM and CNN, they can be derived by modifying the structure.

- *Decoder*: The hidden representation $\boldsymbol{h}$ from the encoder is then mapped back to $\tilde{\boldsymbol{x}}$ through a symmetric multilayer network.

Subsequently, an autoencoder with a simple multilayer network can be easily derived using (2) without considering the past observations, which has been thoroughly investigated in [28]. Driving behavior, however, is a dynamic process in nature which depends on past information [31], [32]. Hence, we take the past observation into consideration using two advanced autoencoders: LSTM-Autoencoder (LSTMAE) and convolutional autoencoder (ConvAE).

1) LSTMAE: In contrast to the simple autoencoder with a multilayer perceptron, the LSTMAE [33] takes the advantage of the temporal information by adding an information selection function, usually as a sigmoid neural net layer, to decide what information is useful and to remember this helpful information, thus transferring the information $\boldsymbol{x}_{(i)}$ at the i^{th}-layer to the representation at the $(i+1)^{\text{th}}$-layer. The LSTMAE comprises four basic equations to achieve this, formulated by

$$z_{(i)} = \sigma(\boldsymbol{w}_z^{\top}\boldsymbol{\ell}_{(i)} + \boldsymbol{u}_z\boldsymbol{x}_{(i-1)} + \boldsymbol{b}_z) \tag{3a}$$

$$y_{(i)} = \sigma(\boldsymbol{w}_y^{\top}\boldsymbol{\ell}_{(i)} + \boldsymbol{u}_y\boldsymbol{x}_{(i-1)} + \boldsymbol{b}_y) \tag{3b}$$

$$\hat{\boldsymbol{x}}_{(i)} = \tanh(\boldsymbol{w}_l^{\top}\boldsymbol{\ell}_{(i)} + \boldsymbol{u}_\ell y_{(i)}\boldsymbol{x}_{(i-1)} + \boldsymbol{b}_\ell) \tag{3c}$$

$$\boldsymbol{x}_{(i)} = (1 - z_{(i)})\boldsymbol{x}_{(i-1)} + z_{(i)}\hat{\boldsymbol{x}}_{(i)} \tag{3d}$$

where $\sigma(\cdot)$ is the sigmoid function, $\boldsymbol{w}_z, \boldsymbol{w}_y, \boldsymbol{w}_\ell$ are the weights, $\boldsymbol{b}_z, \boldsymbol{b}_y, \boldsymbol{b}_\ell$ are the biases, $\boldsymbol{x}_{(i)}$ is the output vector of the LSTM unit. Therefore, given observation $\boldsymbol{x}_{(i-1)}$ for each LSTM unit, we can propagate the output $\boldsymbol{x}_{(i)}$ to next layer.

2) Convolutional Autoencoder (ConvAE): Sharing the same structure with LSTMAE, the ConvAE describes the layer relationship using a convolution operation, instead of using a primary multiply operator.

The encoding and decoding processes can be treated as a procedure for reconstructing observations. Hence, we can learn hidden representation $\boldsymbol{h}$ by minimizing the errors between reconstructed observations $\tilde{\boldsymbol{x}}$ and the origin inputs $\boldsymbol{x}$ with the cost function

$$\theta^* = \arg\min_{\theta} E(\theta) \tag{4}$$

with $E(\theta) = (\boldsymbol{x} - \tilde{\boldsymbol{x}})^{\top}(\boldsymbol{x} - \tilde{\boldsymbol{x}})$ and $\theta = \{\boldsymbol{w}_{(i,i+1)}, \boldsymbol{b}_{(i)}\}_{i=1}^{I}$ for all layers, and I is the number of layers. The discussion above indicates that researchers usually transform the representation learning process into a problem of training autoencoders, then select the output of the encoder $\boldsymbol{h}$ (i.e., the layer of hidden presentation) in feature space $\mathcal{H} \in \mathbb{R}^{r\times 1}$ for each driving encounter $\boldsymbol{x}$ as the expected representations of this driving encounter, with r the dimension of the hidden layer.

B. Distance-Based Measure

Instead of treating the representation learning process as a black box, we could also use a mathematically rigorous way to capture their spatial relationship (e.g., approaching to each other fast or slowly) between two vehicles in a driving encounter. To begin with, we need to define a measure to gauge their geometrical distance. Generally, there are two straightforward ways to achieve this: dynamic time warping and normalized Euclidean distance.

1) Dynamic Time Warping (DTW): Given a driving encounter observation $\boldsymbol{x}$, the DTW [34] aims to measure the geometric relationship of the two-vehicle trajectories over time

$$\boldsymbol{p}^{(1)} = (p_1^{(1)}, \cdots, p_k^{(1)}, \cdots, p_{\bar{K}}^{(1)})$$
$$\boldsymbol{p}^{(2)} = (p_1^{(2)}, \cdots, p_k^{(2)}, \cdots, p_{\bar{K}}^{(2)})$$

where $k = 1, 2, \ldots, \bar{K}$ is sampling index. We define trajectory measure space $\mathcal{P} \in \mathbb{R}^{\bar{K}\times\bar{K}}$, with $p_m^{(1)}, p_n^{(2)} \in \mathcal{P}$ for $m, n \in [1, \cdots, \bar{K}]$, where n and m are the positions of the two vehicles, respectively. The local distance of the positional point of one vehicle to one positional point of the other vehicle is defined and computed by a non-negative function f

$$f : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_{\geq 0} \tag{5}$$

Typically, if $p_m^{(1)}$ and $p_n^{(2)}$ are close to each other, $f(p_m^{(1)}, p_n^{(2)})$ gets a small value, and otherwise $f(p_m^{(1)}, p_n^{(2)})$ attains a large value. Thus calculating the distance of pairwise elements in the two vehicle trajectories $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$ allows to represent the geometry of driving encounter $\boldsymbol{x}$ through a feature matrix $\boldsymbol{F} \in \mathbb{R}^{\bar{K}\times\bar{K}}$

$$\boldsymbol{F}(m, n) := f(p_m^{(1)}, p_n^{(2)}) \tag{6}$$

with the distance measure function f. Here, the Euclidean distance is selected to compute the local distance,

$$f(p_m^{(1)}, p_n^{(2)}) = \| p_m^{(1)} - p_n^{(2)} \|_2 \tag{7}$$

where $\|\cdot\|_2$ indicate the Euclidean distance between two points.

2) Normalized Euclidean Distance (NED): The other efficient and straightforward way to measure the distance of temporal-pairwise positions of two vehicles at time k is to apply an Euclidean distance directly. Thus, given a driving encounter, we can obtain a distance vector $\boldsymbol{f} = [f_1, f_2, \cdots, f_{\bar{K}}]$ to describe the relationship of the two vehicles, where

$$f_k = \|p_k^{(1)} - p_k^{(2)}\|_2 \tag{8}$$

Then we normalized the distance vector as $\boldsymbol{F} = \frac{\boldsymbol{f}}{\max(\boldsymbol{f})}$. Normalization allows us to emphasize the trends in the relative position. Directly applying the unnormalized Euclidean feature vector would cause unexpected clustering results because the value of the initial distance between two vehicles could have a dominant influence on clustering, although two trajectories in one driving encounter have similar moving trends with the other driving encounter. Therefore, we can get the feature $\boldsymbol{F}$ of this driving encounter using DTW and NED.

In addition to DTW and NED, there exist other kinds of distance measures such as LCSS and Edit Distance for Real

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

sequence (EDR) [12]. They discretize trajectories of driving encounter and take account of the number of occuring, but the Euclidean distance between matched segments does not match a predefined spatial threshold. Compared to DTW, both of LCSS and EDR are sensitive to the spatial threshold [12], i.e., a large threshold value indicates a high tolerance of differences in trajectories and vice versa. Therefore, we selected the DTW and NED instead of LCSS and EDR in this paper.

C. Shape-Based Measure

Different from the distance-based measure, the shape-based measure aims to capture the geometric features of two single-vehicle trajectories [5], [12]. The most well-known algorithms are the Hausdorff distance [35], the Fréchet distance [36], and the symmetrized segment-path distance (SSPD) [12]. These methods have registered a high level of performance for describing the geometric features of a single-vehicle trajectory; however, they are not suitable for capturing the relationship between driving encounters consisting of a pair of vehicle trajectories, since their output is a metric and easy to measure the shape-similarity level of individual trajectories but may not work for multi-vehicle trajectories. Even with the same metric value, the driving encounters could be significantly different from each other in terms of shape. Therefore, the shape-based measure is not used in this paper.

D. Summary

According to the above discussion, we select the deep neural network-based autoencoders and the distance-based measure to learn representations of driving encounters. Besides, the autoencoder can capture underlying information through dimension reduction and hence can potentially extract meaningful representations from the results of DTW or NED. There are five possible combinations of deep autoencoders and distance-based measure to learn representations of driving encounters, detailed as follows:

1) DTW: Only using the output of DTW as representations,

$$\mathrm{DTW}: \boldsymbol{x} \xrightarrow{\mathrm{Eq.}(6)} \boldsymbol{F}$$

2) NED: Only using the output of NED as representations,

$$\mathrm{NED}: \boldsymbol{x} \xrightarrow{\mathrm{Eq.}(8)} \boldsymbol{F}$$

3) LSTMAE: Applying LSTMAE to extract the representations of driving encounters,

$$\mathrm{LSTMAE}: \boldsymbol{x} \xrightarrow{\mathrm{LSTMAE}} \boldsymbol{h}$$

4) DTW-ConvAE: Combing ConvAE with DTW to extract representations,

$$\mathrm{DTW-ConvAE}: \boldsymbol{x} \xrightarrow{\mathrm{Eq.}(6)} \boldsymbol{F} \xrightarrow{\mathrm{ConvAE}} \boldsymbol{h}$$

5) NED-LSTMAE: Combine LSTMAE with NED to extract representations,

$$\mathrm{NED-LSTMAE}: \boldsymbol{x} \xrightarrow{\mathrm{Eq.}(8)} \boldsymbol{F} \xrightarrow{\mathrm{LSTMAE}} \boldsymbol{h}$$

TABLE I
INPUT AND OUTPUT OF REPRESENTATION LEARNING APPROACHES

Method	Input observations	Output features
DTW	$\boldsymbol{x} \in \mathbb{R}^{K\times 1}$	$\boldsymbol{F} \in \mathbb{R}^{K\times K}$
NED	$\boldsymbol{x} \in \mathbb{R}^{K\times 1}$	$\boldsymbol{F} \in \mathbb{R}^{K\times 1}$
LSTMAE	$\boldsymbol{F} \in \mathbb{R}^{K\times 1}$	$\boldsymbol{h} \in \mathbb{R}^{r\times 1}$
DTW-ConvAE	$\boldsymbol{F} \in \mathbb{R}^{K\times K}$	$\boldsymbol{h} \in \mathbb{R}^{r\times 1}$
NED-LSTMAE	$\boldsymbol{F} \in \mathbb{R}^{K\times 1}$	$\boldsymbol{h} \in \mathbb{R}^{r\times 1}$

To understand this efficiently, we display and compare the input and output for each method of extracting representations in Table I. For the ease of representation, we denoted $\mathcal{F}_{\boldsymbol{x}_i}$ as the extracted representation $\boldsymbol{F}$ or $\boldsymbol{h}$ of driving encounter $\boldsymbol{x}_i$ for all approaches in Table I, where K is the dimension of inputs and r is the dimension of the learned representations.

IV. CLUSTERING

This section introduce clustering algorithms for the learning representations. We first detail the clustering method and then introduce the performance evaluation criteria of clustering results. Time-series trajectories clustering is very ubiquitous [37], and many clustering algorithms have been developed to solve related problems [38]. The learned representations of driving encounters restrict clustering method selection. Given the extracted representation of driving encounters in Table I, we can classify them, i.e., gather associated driving encounters into groups. The learned representations are usually in a high dimension, which makes it practically intractable for specific clustering approaches such as DBSCAN (i.e., density-based spatial clustering of applications with noise). In this paper, we prefer the k-means clustering (k-MC) [39] for simplicity and scalability.

During the clustering procedure, we have limited prior knowledge about the number of groups. Therefore, we need a criterion to evaluate the clustering performance and consequently select an appropriate cluster number. We aim to gather driving encounters with similar characteristics into one group. Hence, the quality of clustering results can be assessed by checking the between and within cluster variances [40]. In other words, the variance of the elements in the same group should be as small as possible, while the variance between groups should be as large as possible. According to references [12], [37], [38], we define the within-cluster (WC) variance and between-cluster (BC) variance, which requires the computation of a mean value of extracted features. Here, we used the Calinksi-Harabasz index [40] to evaluate the clustering performance as for real world data we do not have the ground truth of clustering. This evaluation index can be easily implemented by using a Python tool called `scikit-learn`. Directly computing the mean of driving encounters is intractable since it is inconceivable and meaningless to calculate the mean of trajectories in driving encounters. Instead of using the multi-vehicle trajectories to evaluate the BC and WC variances directly, we computed the mean of the extracted representations (denoted as $\bar{\mathcal{F}}_{\mathcal{X}}$) by averaging all extracted representations of driving encounters belong to cluster $\mathcal{X}$. Let $\mathcal{X}_j$ be the set of driving encounters in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

j-th clusters. Assuming that we obtain J clusters for N driver encounters, then the BC variance and WC variance are computed by

$$BC = \frac{1}{J-1}\sum_{j=1}^{J}\mathcal{D}(\bar{\mathcal{F}}_{\mathcal{X}}, \bar{\mathcal{F}}_{\mathcal{X}_j}) \tag{9a}$$

$$WC = \frac{1}{N-J}\sum_{j=1}^{J}\sum_{i}^{|\mathcal{X}_j|}\frac{1}{|\mathcal{X}_j|}\mathcal{D}(\mathcal{F}_{\boldsymbol{x}_{j,i}}, \bar{\mathcal{F}}_{\mathcal{X}_j}) \tag{9b}$$

where $\mathcal{D}(\cdot,\cdot)$ is the Euclidean distance measure, $|\mathcal{X}_j|$ represents the number of driving encounters in the cluster $\mathcal{X}_j$, $\mathcal{F}_{\boldsymbol{x}_{j,i}}$ is the feature representation of the i-th driving encounter in the j-th cluster, and $\bar{\mathcal{F}}_{\mathcal{X}_j}$ is the mean of the feature representations of all driving encounters in the j-th cluster. Our goal is to maximize the BC variance and minimize the WC variance; however, the BC and WC variances of representations extracted by different approaches are usually in different scales, which makes it meaningless to compare their BC and WC values directly. In order to make performance comparison tractable, we normalized the BC and WC variances by defining their relative metrics as follows:

$$\lambda_{BC} = \frac{BC}{WC + BC} \tag{10}$$

$$\lambda_{WC} = \frac{WC}{WC + BC} \tag{11}$$

In this way, the BC and WC variances between approaches become comparable by scaling their values between [0, 1] with respect to each approach. A large value of λ_{BC} indicates a large distance between clusters and otherwise, indicates a small distance between clusters.

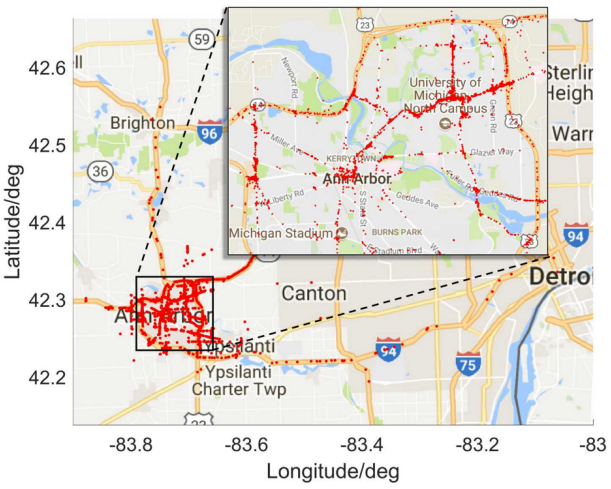


Fig. 5. Scatter plot of driving encounters in Ann Arbor on the Google Map.

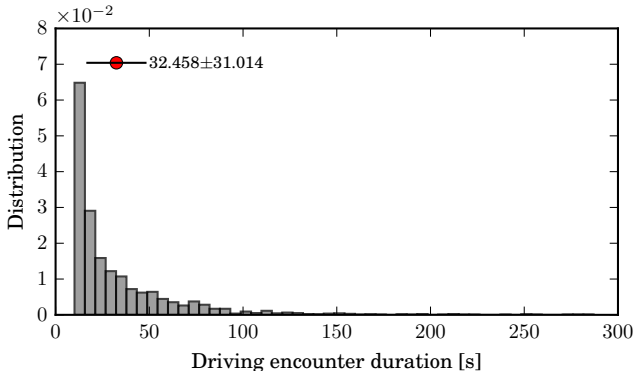


Fig. 6. Distribution of origin driving encounter duration in seconds.

V. Experiment and Data Collection

A. Data Collection and Analysis

All the driving encounter data were collected from naturalistic settings supported by the University of Michigan Safety Pilot Model Development (SPMD) program [41]. This SPMD database was collected by the University of Michigan Transportation Research Institute (UMTRI) and provided driving data logged in the last three years in Ann Arbor area, covering about 3,500 equipped vehicles and 6-million trips in total. The onboard GPS start to collect latitude and longitude information of each vehicle while igniting the equipped vehicle. The data were collected at a sampling frequency of 10 Hz.

We searched the dataset of 100,000 trips, collected from 1900 vehicles with 12-day runs. The trajectory information we extracted includes latitude, longitude, the speed of the vehicles. The selection range was restricted to an urban area with the latitude in (-83.82, -83.64) and the longitude in (42.22, 42.34), as shown in Fig. 5. The vehicle encounter refers to as the scenario where two vehicles are close to each other, and the distance in the Euclidean space between them is smaller than 100 m. The dots indicate the position of the vehicle at every sample time. After querying from the SPMD database, we got 49,998 such vehicle encounters. In this paper, we mainly focus on the trajectory length longer than 10 s, which is suitable for analysis and applications. Here, we set the duration limitation for each selected driving encounter, but this would not affect the generalization capability of our proposed method. Finally, we obtained 2,568 driving encounters for experimental validation which fit these criteria. Fig. 6 displays the distribution of time duration of all origin driving encounters.

B. Experiment Procedure and Settings

1) Autoencoders: In order to make feature extraction efficiently, we used the re-sampling method (mentioned in Section II-B) through an interpolation process to unify the driving encounter into equal length $\bar{K}$. Fig. 6 displays that the length of driving encounters was diverse, and the mean value of them is around 30 s. In order to facilitate the training procedure, we empirically unified each driving encounter to a fixed length of 200 sample points using linear interpolation. Here, we selected the unified length of 200 with the fact that a small number of unified samples will reduce model accuracy while a high number of unified samples will increase computational burden without significant model performance improvement. We set all autoencoders with a symmetric structure based on our experiences, and more specifically,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

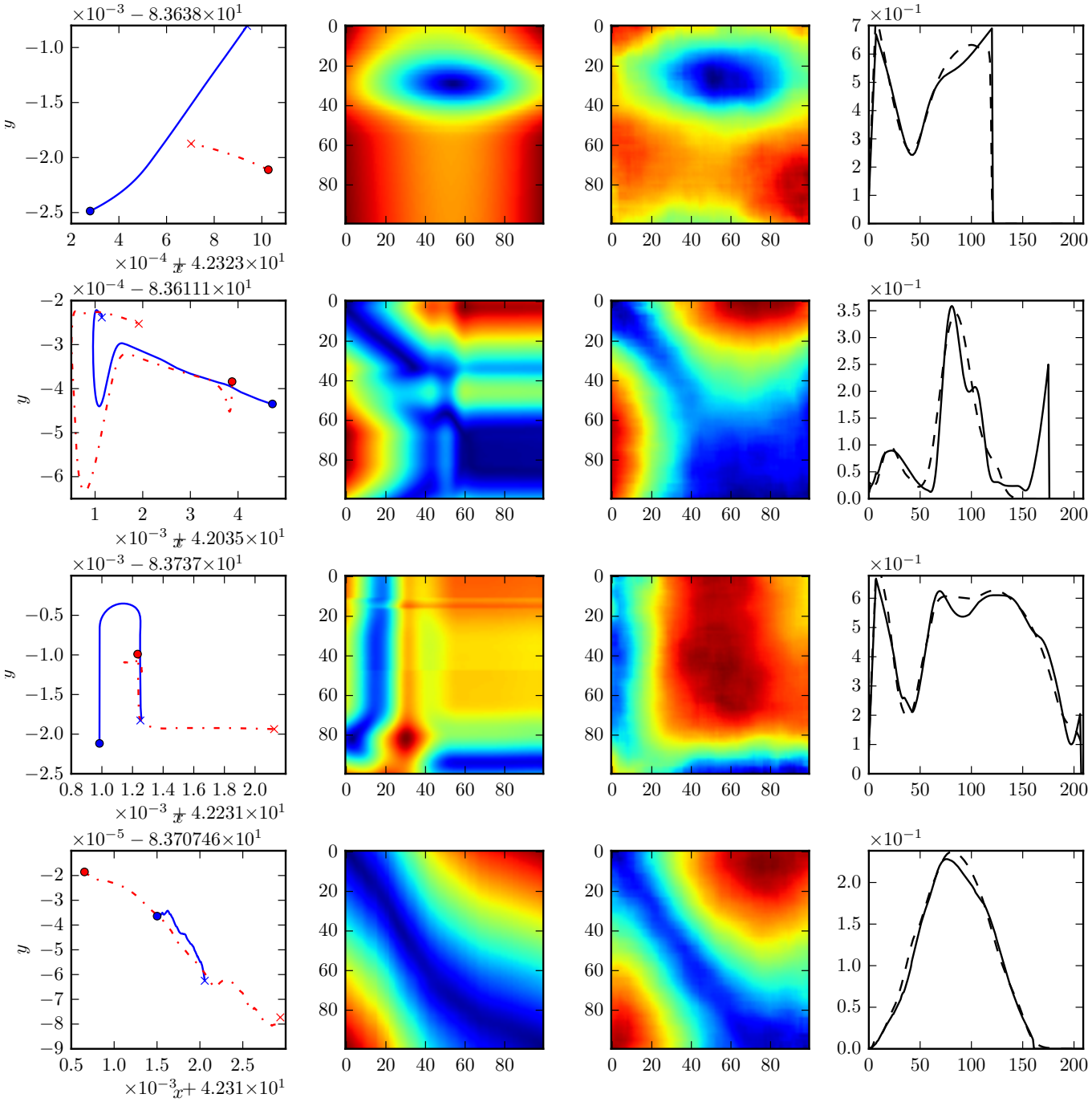


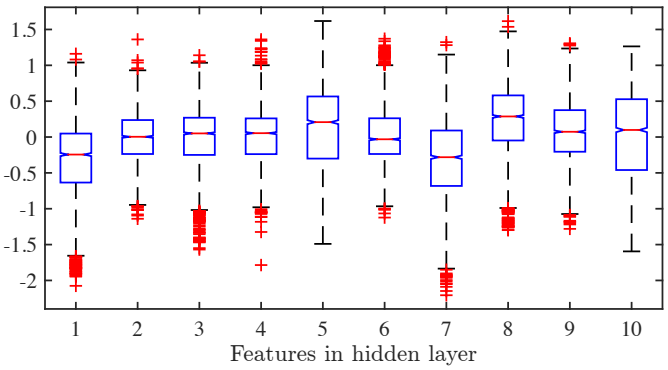
Fig. 7. Examples of learning representations by different approaches. Left: Raw driving encounter GPS trajectories with start points (i.e., marked as dot) and end points (i.e., marked as cross). Mid-left: Extracted representation using DTW. Mid-right: Reconstructed representation of DTW using ConvAE. Right: Extracted representation using normalized Euclidean distance (NED) of two trajectories and its reconstructed representation from LSTMAE.

- LSTMAE: In this paper, we designed a five-layer ($I = 5$) LSTMAE for learning the representations, where the third layer (i.e., the middle layer) is extracted as the hidden representation of driving encounters. The weights for each layer were initialized using random numbers between 1 and -1 following a uniform distribution. The latent dimension (i.e., hidden layer) was set as 10, i.e., $\boldsymbol{h} \in \mathbb{R}^{r\times 1}$ with $r = 10$.
- ConvAE: A five-layer ConvAE was designed to extract underlying features from outputs of DTW. The dimension of its hidden layer was also set as 10, i.e., $\boldsymbol{h} \in \mathbb{R}^{r\times 1}$ with $r = 10$. Applying the ConvAE to the high-dimension DTW feature matrix can significantly reduce the computation burden as the ConvAE can be typically used for dimensionality reduction.

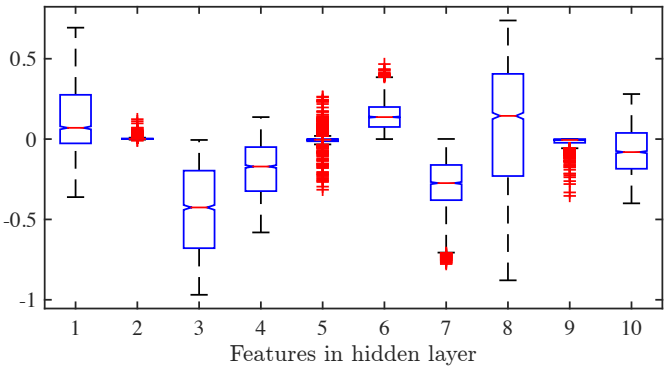
After learning the hidden feature $\mathcal{F}_{x_i}$ of driving encounter $\boldsymbol{x}_i$ using autoencoders, we then applied the k-means clustering (k-MC) approach to these extracted hidden representations.

2) DTW: Similar to autoencoders, the length of driving encounters was unified to 100 for DTW. Thus we can compute

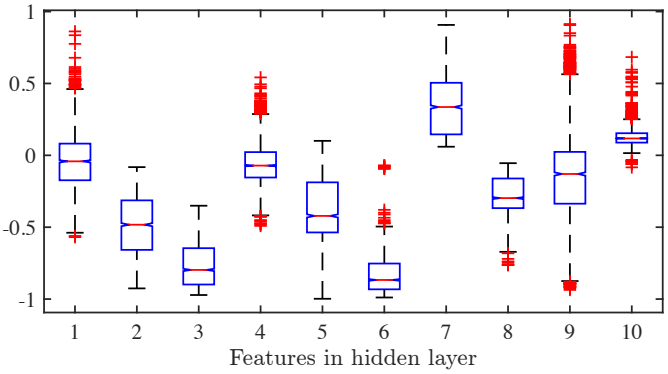
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles



(a) DTW-ConvAE



(b) NED-LSTMAE



(c) LSTMAE

Fig. 8. Statistical results of extracted hidden features $\boldsymbol{h} \in \mathbb{R}^{10\times 1}$ for 2,568 driving encounters using three autoencoders: (a) DTW-ConAE, (b) NED-LSTMAE, and (c) LSTMAE.

the representation $\mathcal{F}_{\boldsymbol{x}} \in \mathbb{R}^{100\times 100}$ using (7) for each driving encounter. Then we apply the clustering algorithms to all $\mathcal{F}_{\boldsymbol{x}}$.

3) NED: For the NED approach, the representation can be directly computed through (8) based on the unified driving encounters. Then, the output can be directly used for clustering and feeding into LSTMAE to obtain a low dimension representation (i.e., NED-LSTMAE).

For all learned representations, we do have limited prior knowledge of what values of $k \in \mathbb{N}^+$ should be set. In order to determine k, we applied clustering algorithms to different k and computed their performance metrics λ_{BC} and λ_{WC}.

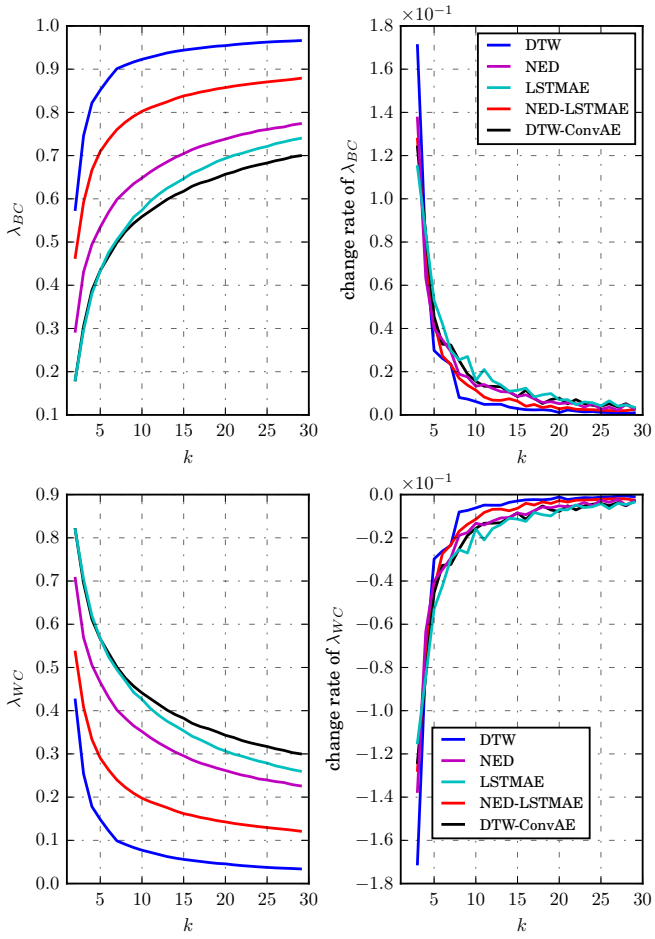


Fig. 9. The performance of BC (top) and WC (bottom) distances using five representation extraction approaches (i.e., DTW, NED, LSTMAE, DTW-ConvAE, and NED-LSTMAE) with k-MC.

VI. Result Discussion and Analysis

This section will present and analyze the experiment results, including representation extraction results and clustering performance evaluation and comparison.

A. Representation Learning and Analysis

Fig. 7 displays four examples of learned representations using DTW and NED and the associated reconstructed results of using LSTMAE and ConvAE. The second column displays the DTW representations of associated driving encounter trajectories (the first column), and the third column displays the reconstruction results of using ConvAE from the hidden representation, $\boldsymbol{h}$. The second and third columns show similar feature matrix values (represented by color), indicating that the ConvAE can completely reconstruct the representations of DTW. On the other hand, the fourth column represents the NED results (solid line) and the reconstruction of NED outputs (dash line), and we can see that they match well with each other. In summary, the representations in the hidden layers of autoencoders (i.e., ConvAE and LSTMAE) can represent the associated driving encounters.

1) Using DTW and NED: Fig. 7 shows examples of the learned representations using DTW and NED. It is known that both DTW and NED methods can capture the distance information of two vehicle trajectories in one driving encounters. In the extracted representations of using DTW, deep red indicates a large distance of two vehicles and deep blue indicates a small distance. The DTW can capture the spatial information of all positions and the dynamic information over the whole trajectory since it computes the distance of trajectories over time, while the NED does not capture the dynamic information over temporal space. In addition, Fig. 7 presents the associated reconstructed outputs of DTW and NED. Experimental results show that the ConvAE and LSTMAE captured most underlying information of the extracted representations using DTW and NED, respectively.

2) Using Autoencoders: Each driving encounter results in a ten-dimensional representation vector $\boldsymbol{h} \in \mathbb{R}^{10\times 1}$ from the designed autoencoders (i.e., ConvAE and LSTMAE). Fig. 8 shows box plots of hidden layers in different autoencoders for all driving encounters. For each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. We found that DTW-ConvAE obtains less recognizable hidden representations; that is, distributions between each element of representation $\boldsymbol{h}$ do not have significant differences, compared to NED-LSTMAE and LSTMAE. All elements in extracted representations using DTW-ConvAE have similar median values almost around zero and similar ranges of [25th, 75th] percentiles. For the NED-LSTMAE and LSTMAE methods, both of their hidden representations are recognizable, with median values and ranges of [25th, 75th] percentiles are sparsely distinctive.

B. Clustering Results and Analysis

Based on the extracted representations, Fig. 9 compares all approaches to cluster driving encounters by showing the scaled within-cluster and between-cluster metrics (λ_{BC} and λ_{WC}) and their change rates over the number of clusters k. We found that increasing the number of clusters would decrease the within-cluster distance while increasing the between-cluster distance. According to their change rate of λ_{BC} or λ_{WC}, when the number of clusters is close to 10, their performance metrics would converge, implying that $k = 10$ is the preferred selection. As we discussed before, the goal of clustering is to put homogeneous driving encounters into groups while maximizing the between-cluster distance and minimizing the within-cluster distance. Fig. 9 indicates that the DTW-kMC outperforms other counterparts, with $\lambda_{BC} = 0.923$ at $k = 10$. This is because the DTW can capture the interaction features of two vehicles at different time steps. For example, the elements close to the main diagonal of the DTW feature matrix can represent how one vehicle would move at the current moment after observing the positions of the other vehicle at the last moment. However, the autoencoder combined with DTW obtains the lowest performance because a 10-dimensional vector (i.e., the dimension of hidden layer in autoencoders) may not sufficiently represent a high-dimensional feature matrix.

TABLE II
EFFICIENCY COMPARISON OF REPRESENTATION EXTRACTION APPROACHES

Methods	Computing time	Unit
NED	101.41	second
DTW	120.77	second
LSTMAE	≈ 10	hour
ConvAE	≈ 5	hour

For all autoencoders at $k = 10$, the NED-LSTMAE-kMC obtains the best performance with $\lambda_{BC} = 0.803$, compared to LSTMAE-kMC with $\lambda_{BC} = 0.574$ and DTW-ConvAE-kMC with $\lambda_{BC} = 0.559$. This can be explained according to Fig. 8: The DTW-ConvAE-kMC (top in Fig. 8) obtains the worst performance as its extracted representations of driving encounters are not such recognizable and distinctive, compared with the other two approaches. For the distance-based measure, the DTW-kMC outperforms the NED-kMC with $\lambda_{BC} = 0.648$, because the DTW can capture the dynamic features over time and the distance features of two trajectories, but the NED cannot.

Fig. 10 visualizes the results of all clustered driving encounters with their GPS data. We only show the results with the optimal number of clusters $k = 10$ because we have seen in Fig. 9 that a plateau can be observed on the change rate of within-cluster criteria λ_{BC} with respect to the number of cluster starting at cluster sizes of 10 for all approaches. Fig. 11 lists the amount of driving encounters in each cluster. It can be found that some clusters covering very common driving encounter behaviors (e.g., cluster #2, cluster #7, and cluster #9) and some clusters representing rare driving encounter behavior (e.g., cluster #4 and cluster#10) can be detected. For cluster #1 (red) and cluster #10 (gray), they represent two typical driving encounter scenarios, i.e., car-following on highways and driving with the opposite directions, respectively. For the most occurred driving encounter scenario (cluster #2, cluster #7, and cluster #9), most of them are recorded at intersections (i.e., one car moves straight and the other car moves straight and then takes a right turn with different directions) with a short duration, compared to cluster #1 and cluster #10. For the cluster #4, it represents the scenarios where one car keeps stationary and the other car keeps moving, while the cluster #5 represents a kind of car-following behaviors at the intersection or on the highway ramp.

C. Computing Efficiency Analysis

In order to evaluate the conservativeness of the algorithm and its applicability to real-time applications, its computational costs when implemented on a standard laptop computer are presented, and a comparison with other counterparts is given. All autoencoders were built using Python in Keras[1] with a tensorflow backend and all distance-based representation extraction algorithms were also programmed using Python. Table II presents the average computation time for extracting representations on a standard laptop computer with an Intel

[1]https://blog.keras.io/building-autoencoders-in-keras.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2020.2973550, IEEE Transactions on Intelligent Vehicles

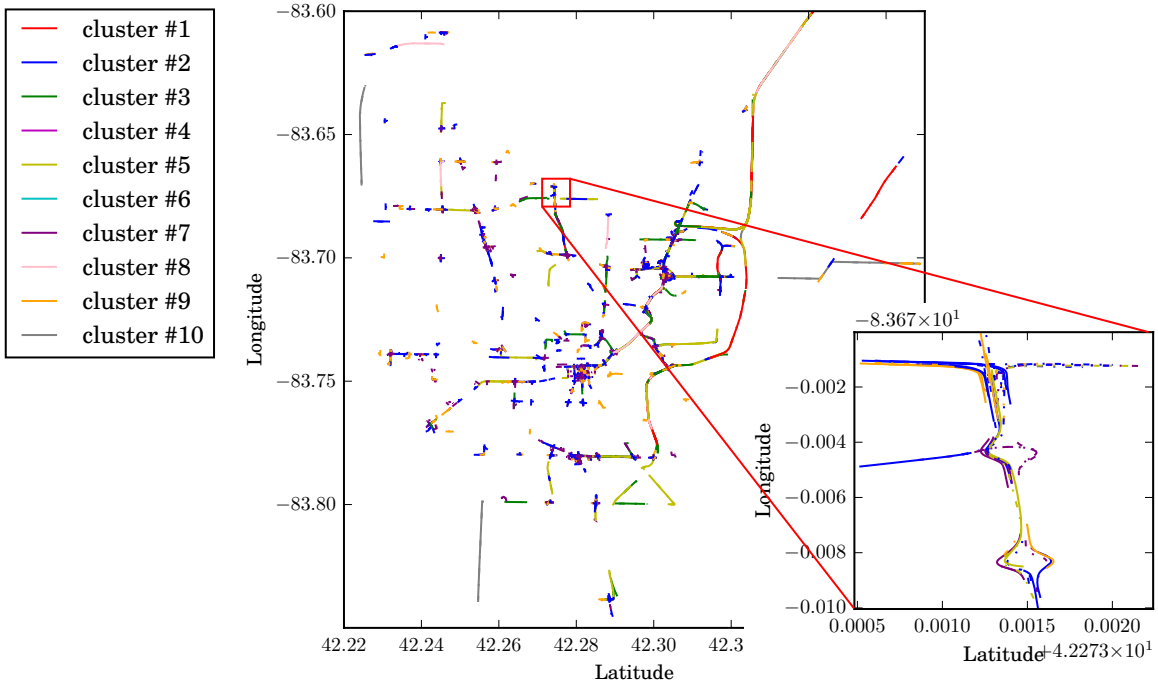


Fig. 10. GPS data visualization of the clustering results using DTW-kMC with $k = 10$.

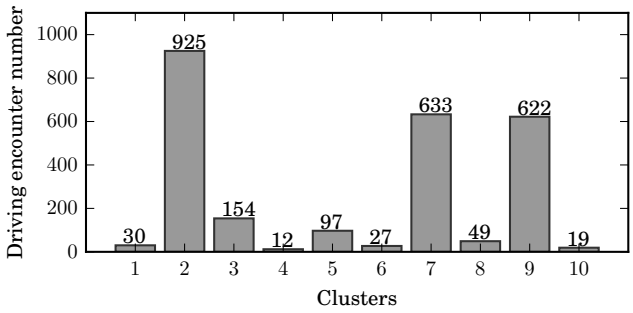


Fig. 11. Clustering results of using DTW-kMC with $k = 10$.

Core i7 running at 2.5 GHz, with 16 GB of RAM. It can be seen that the NED and DTW algorithms can extract feature much faster than two others, only with 120 s for 2568 driving encounters. However, the autoencoders with neural nets require a few hours to extract representations.

D. Further Discussion

This paper presents a comprehensive investigation of driving encounter classification through five approaches, which consists of two types: deep learning-based and distance-based. The distance-based method outperforms all other counterparts. However, some challenges still exist in our developed approaches and discussed as follows.

1) Layers of Autoencoders: For the deep learning-based approaches, we empirically set their hyperparameters, like the number of layers of decoder and encoder and the number of nodes in each layer. The hyperparameter selection in our paper mainly concerns two aspects: computational cost and model performance. Autoencoder with large numbers of layers could enhance model performance but at a significantly high computational cost. Therefore, in this paper, we selected a moderate number of layers to learn a hidden representation of driving encounters.

2) Contextual Road Information: This paper mainly utilized the multi-vehicle trajectories (GPS signals) without utilizing other information since GPS data is easy to obtain at a low cost such as via mobile-phone and equipped localization sensors. Our experiment validation did not consider other information such as contextual road information and vehicle speed, but it might extend our developed approach to other driving cases with high-dimensional time-series data. For instance, if more road context information could be obtained such as highway and intersections (T-shape, Y-shape, and cross-shape, etc.), our developed approaches can help get insights into the diversity of driving encounters. Therefore, considering contextual road information will be one of our future work. Moreover, the feature selection can influence the clustering results, namely, adding more feature information (e.g., vehicle speed, contextual traffic, the number of agents in driving encounters, etc.) as inputs could change the final number of clusters.

E. Multi-Vehicle Interactions

Validation and demonstration of our proposed framework towards clustering connected vehicle trajectories in this paper (Fig. 2) is for two-vehicle interactions. The key to achieving this is to select representative features which should capture the expected characteristics. This framework can be directly applied to multi-vehicle interaction trajectories clustering by

carefully selecting the features through representation learning techniques. For example, the feature of mutli-vehicle interactions at each moment can be represented by the hidden layers of autoencoders, or velocity fields with stochastic processes such as Gaussian process [42], [43].

VII. Conclusion

This paper proposed a generic two-layer framework of clustering naturalistic driving encounters that consists of multi-vehicle trajectories. Two kinds of representation learning – deep autoencoders and distance-based were developed and evaluated. Experimental results show that our proposed framework can gather homogeneous driving encounters into groups considering the spatiotemporal relationship between vehicles. Besides, the dynamic time warping (DTW) approach with k-means clustering method outperforms other counterparts, in terms of both clustering performance and computational burden. We finally evaluated the performance of each proposed approach and confirmed that when we are only concerned with the vehicle trajectories, an acceptable and preferable cluster number is 10 in our database. These clustering results could provide insights into interactive driver behaviors in different scenarios and benefit friendly and efficient decision-making policy design for intelligent vehicles by recognizing and classifying interactions in the multi-model setting [44], [45].

Acknowledgment

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.

[2] J. Taylor, X. Zhou, N. M. Rouphail, and R. J. Porter, "Method for investigating intradriver heterogeneity using vehicle trajectory data: A dynamic time warping approach," *Transportation Research Part B: Methodological*, vol. 73, pp. 59–80, 2015.

[3] 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.

[4] L. Sun, W. Zhan, Y. Hu, and M. Tomizuka, "Interpretable modelling of driving behaviors in interactive driving scenarios based on cumulative prospect theory," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, New Zealand: IEEE, 2019, pp. 4329–4335.

[5] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Destination prediction by trajectory distribution-based model," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[6] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for video Technology*, vol. 18, no. 11, pp. 1544–1554, 2008.

[7] Z. Sun, P. Hao, X. J. Ban, and D. Yang, "Trajectory-based vehicle energy/emissions estimation for signalized arterials using mobile sensing data," *Transportation Research Part D: Transport and Environment*, vol. 34, pp. 27–40, 2015.

[8] Z.-J. Chen, C.-Z. Wu, Y.-S. Zhang, Z. Huang, J.-F. Jiang, N.-C. Lyu, and B. Ran, "Vehicle behavior learning via sparse reconstruction with $\ell_2 - \ell_p$ minimization and trajectory similarity," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 236–247, 2017.

[9] T. Appenzeller, "The scientists' apprentice," *Science*, vol. 357, no. 6346, pp. 16–17, 2017.

[10] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2017.

[11] Z. Feng and Y. Zhu, "A survey on trajectory data mining: Techniques and applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.

[12] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3306–3317, 2016.

[13] D. Yao, C. Zhang, Z. Zhu, Q. Hu, Z. Wang, J. Huang, and J. Bi, "Learning deep representation for trajectory clustering," *Expert Systems*, DOI: 10.1111/exsy.12252.

[14] H. Zhao, C. Wang, Y. Lin, F. Guillemard, S. Geronimi, and F. Aioun, "On-road vehicle trajectory collection and scene-based lane change analysis: Part I," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 192–205, 2017.

[15] W. Yao, Q. Zeng, Y. Lin, D. Xu, H. Zhao, F. Guillemard, S. Geronimi, and F. Aioun, "On-road vehicle trajectory collection and scene-based lane change analysis: Part II," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 206–220, 2017.

[16] M. Y. Choong, L. Angeline, R. K. Y. Chin, K. B. Yeo, and K. T. K. Teo, "Modeling of vehicle trajectory clustering based on lcss for traffic pattern extraction," in *Automatic Control and Intelligent Systems (I2CACIS), 2017 IEEE 2nd International Conference on*. Kota Kinabalu, Malaysia: IEEE, 2017, pp. 74–79.

[17] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE transactions on intelligent transportation systems*, vol. 11, no. 3, pp. 647–657, 2010.

[18] J. Melo, A. Naftel, A. Bernardino, and J. Santos-Victor, "Detection and classification of highway lanes using vehicle motion trajectories," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 2, pp. 188–200, 2006.

[19] S. Chauvin, "Hierarchical decision-making for autonomous driving," DOI: 10.13140/RG/2.2.24352.43526, 2018.

[20] P. Zhao, K. Qin, X. Ye, Y. Wang, and Y. Chen, "A trajectory clustering approach based on decision graph and data field for detecting hotspots," *International Journal of Geographical Information Science*, vol. 31, no. 6, pp. 1101–1127, 2017.

[21] J. Wang, C. Wang, X. Song, and V. Raghavan, "Automatic intersection and traffic rule detection by mining motor-vehicle gps trajectories," *Computers, Environment and Urban Systems*, vol. 64, pp. 19–29, 2017.

[22] X. Zhan, Y. Zheng, X. Yi, and S. V. Ukkusuri, "Citywide traffic volume estimation using trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 272–285, 2017.

[23] J. Kim and H. S. Mahmassani, "Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories," *Transportation Research Part C: Emerging Technologies*, vol. 59, pp. 375–390, 2015.

[24] W. Zhang and W. Wang, "Learning v2v interactive driving patterns at signalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 108, pp. 151–166, 2019.

[25] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, vol. 41, no. 6, pp. 1367–1382, 2017.

[26] S. Yang, W. Wang, Y. Jiang, J. Wu, S. Zhang, and W. Deng, "What contributes to driving behavior prediction at unsignalized intersections?" *Transportation research part C: emerging technologies*, vol. 108, pp. 100–114, 2019.

[27] F. T. Pokorny, K. Goldberg, and D. Kragic, "Topological trajectory clustering with relative persistent homology," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. Stockholm, Sweden: IEEE, 2016, pp. 16–23.

[28] S. Li, W. Wang, Z. Mo, and D. Zhao, "Clustering of naturalistic driving encounters using unsupervised learning," *arXiv preprint arXiv:1802.10214*, 2018.

[29] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," *arXiv preprint arXiv:1802.06971*, 2018.

[30] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 3880–3887.

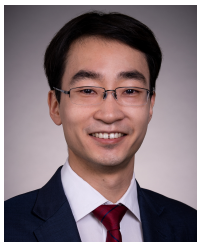
[31] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[32] M. C. Nechyba and Y. Xu, "Stochastic similarity for validating human control strategy models," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 437–451, 1998.
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.
[35] A. A. Taha and A. Hanbury, "An efficient algorithm for calculating the exact hausdorff distance," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 11, pp. 2153–2163, 2015.
[36] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk, "Fréchet distance for curves, revisited," in *European Symposium on Algorithms*. Springer, 2006, pp. 52–63.
[37] T. W. Liao, "Clustering of time series data: A survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.
[38] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.
[39] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881–892, 2002.
[40] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. PéRez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.
[41] W. Wang, C. Liu, and D. Zhao, "How much data are enough? a statistical approach with case study on longitudinal driving behavior," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 2, pp. 85–98, 2017.
[42] C. Zhang, J. Zhu, W. Wang, and D. Zhao, "A general framework of learning multi-vehicle interaction patterns from video," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, New Zealand: IEEE, 2019, pp. 4323–4328.
[43] Y. Guo, V. V. Kalidindi, M. Arief, W. Wang, J. Zhu, H. Peng, and D. Zhao, "Modeling multi-vehicle interaction scenarios using gaussian random field," *IEEE Intelligent Transportation Systems Conference*, 2019.
[44] S. Gopalswamy and S. Rathinam, "Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China: IEEE, 2018, pp. 986–992.
[45] R. D. d. L. Torres, M. Molina, and P. Campoy, "Survey of bayesian networks applications on unmanned intelligent autonomous vehicles," *arXiv preprint arXiv:1901.05517*, 2019.

Wenshuo Wang (S'15-M'18) received his Ph.D. degree in mechanical engineering from the Beijing Institute of Technology (BIT) in June, 2018. He is now working as a postdoc at Carnegie Mellon University (CMU), Pittsburgh, PA. He also worked as a Research Scholar in the Department of Mechanical Engineering, University of California at Berkeley (UCB) from September 2015 to September 2017 and in the Department of Mechanical Engineering, University of Michigan (UM), Ann Arbor, from September 2017 to July 2018. His research interests include nonparametric Bayesian learning, driver models, human-vehicle interactions, and the recognition and application of human driving characteristics.

Aditya Ramesh is a Master student with the Department of Electrical Engineering and Compuer Science (EECS), University of Michigan, Ann Arbor, MI, USA.

His research interests include unsupervised learning and autonomous driving.



Ding Zhao received his Ph.D. degree in 2016 from the University of Michigan, Ann Arbor. He is currently an Assistant Professor at Department of Mechanical Engineering, Carnegie Mellon University. His research focuses on the intersection of robotics, machine learning, and design, with applications on autonomous driving, connected/smart city, energy efficiency, human-machine interaction, cybersecurity, and big data analytics.