Integrating Heterogeneous Sources for Learned Prediction of Vehicular Data Consumption

Andi Zang*, Xiaofeng Zhu†, Ce Li ‡, Fan Zhou‡ and Goce Trajcevski§
*Department of Computer Science
Northwestern University, Evanston, IL/USA
Email: {andi.zang}@u.northwestern.edu
†Microsoft, Redmond, WA/USA
Email: {xiaofzhu}@microsoft.com
‡ School of Information and SW Engineering
University of Electronic Science and Technology, Chengdu, PR China
Email: {ce.lc}@std.uestc.edu.cn, {fan.zhou}@uestc.edu.cn
§ Department of Electrical and Computer Engineering
Iowa State University, Ames, IA/USA
Email: {gocet25}@iastate.edu

2022 23rd IEEE International Conference on Mobile Data Management (MDM) | 978-1-6654-5176-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/MDM55031.2022.00029

***Abstract*—In addition to the multiple sensors to measure parameters that can be used to improve both safety and efficiency, modern vehicles also gather information about external data (e.g., traffic conditions, weather) which, if properly used, could further improve the overall trip experience. Specifically, when it comes to navigation, one source that can provide increased context awareness, especially for autonomous driving, are the High Definition (HD) maps, which have recently witnessed a tremendous growth of popularity in vehicular technology and use. As they are limited to a particular geographic area, different portions need to be downloaded (and processed) on multiple occasions throughout a given trip, along with the other data from other internal and external sources.**

In this paper, we provide an effective deep learning approach for the recently introduced problem of Predicting Map Data Consumption (PMDC) in the future time instants for a given trip. We propose a novel methodology that integrates multiple data sources (road network, traffic, historic trips, HD maps) and, for a given trip, enables prediction of the map data consumption. Our experimental observations demonstrate the benefits of the proposed approach over the candidate baselines.

***Index Terms*—Mobile Data Consumption, Prediction, High Definition Maps.**

I. Introduction and Motivation

Ensuring driving safety is a paramount in the autonomous driving industry, and the combining of on-board real-time sensing techniques and (external) knowledge based "verification" algorithms is a belt-and-braces approach to achieve the objective [1]–[3]. Low-level real-time perception systems involving cameras and LiDAR, accompanied with machine learning [4], have shown impressive performance in well-controlled environments and scenarios. However, even more functionalities can be realized, such as a higher level of assisted driving, improvement of fuel/energy consumption and driving experience/comfort, with the help of High Definition (HD) maps [5],

Among other applications and systems, HD maps have been used in vehicle self-localization [6], however, their notable feature is that they are of enormous data size. They store representations of road objects such as lane boundaries, pole-like objects, occupancy grids and other entities, which are also known as road "furniture". Nowadays, the HD maps can easily contain over a thousand voxels (highway scenario) or even tens of thousands of voxels (urban scenario) per road meter at a higher resolution, in contrast to dozens of points per road link in conventional maps [7]. Complementary to this, the map-based solutions to vehicular tasks such as self-localization [6], visualization and micro motion planning/adjustment, are computationally expensive. These two factors (data size and computational complexity) result in HD maps being the largest consumer of processing power and transmission bandwidth, from server end, through network, to vehicle/user end.

In recent years, numerous companies have started to send out their experimental AVs (Autonomous Vehicles) on public roads. For example, in California, roughly 650 Avs have completed trips of cumulative length of $2,855,739$ and $1,955,201$ miles in 2019 and 2020 respectively [8]. Considering the market size of the "conventional" (i.e., without any assisted driving features) vehicles, increasing investment in [9] and the continuous growth of AVs [10], in the foreseeable future, vehicles with high level assisted driving functions are likely to dominate the market. Currently, experimental/testing vehicles equipped with high performance on-board hardware can easily handle the load of both storage and computation – but with an overhead of high cost. When AVs become commercialized and operated as daily drivers, HD maps (and real-time/live maps) streaming will cause extremely heavy communication-load to the network, linearly increasing with the number of on-line vehicles on the roads.

Hence, optimizing the use of the HD maps data is of primary importance in many AV tasks. The most straightforward and widely used solution is to shrink/compress the map data, in order to use the bandwidth more efficiently [11], [12]. In general, HD maps objects (i.e., furniture) are represented as voxels over an underlying grid [13], and are potentially compatible

2375-0324/22/$31.00 ©2022 IEEE
DOI 10.1109/MDM55031.2022.00029

with hierarchical structures. Furniture can be downloaded on-demand and at different resolutions, depending on the system configurations (e.g., hardware limitation and safety requirements) of particular use cases.

Brute-force downloading all the data, even with a high compression rate, does not solve the problem. In most cases, on-board hardware struggles with handling expensive algorithms which, in addition to delaying the progress of other applications, may also increase the decision response time, which raises safety concerns.

If the Map Data Consumption (MDC) – in total, or even at certain time-intervals – can be predicted, the system will have more time to arrange the size of streaming data and, just as importantly, couple it with other external variables (e.g., weather, traffic updates, etc.). This would enable designing more efficient and reliable/robust context-aware strategies for HD maps data download. As a specific example, driving during a heavy rain and in areas with pedestrians would require higher resolution HD maps to be downloaded, for safer navigation in AVs.

We have recently introduced the problem of Prediction of Map Data Consumption (PMDC) in [14] and proposed a naïve solution. In this paper, we incorporate more heterogeneous datasets, devise novel methodologies for data integration and provide a deep learning based architecture for effective solution to PMDC. Our main contributions can be summarized as follows:

- We refine the definition of MDC problem and provide more rigorous formalism to specify the (integration of the) input data and the output of the predictions.
- We propose a novel Neural Network (NN) structure – a hybrid of GNN and LSTM – that takes heterogeneous data sources (graph, trips and tiles) and combines their encodings.
- We propose a GNN-based solution to PMDC problem, which learns not only local sequential properties (from a trip), but spatial and temporal information from adjacent edges.
- We provide experimental evaluation over a synthetic dataset, demonstrating that our solution provides significant improvements over baselines.

In the rest of this paper, in Sec. II we review the related works, with a note that PMDC is a relatively novel problem. In Sec. III, we give a detailed overview and formalize the definition of the PMDC, along with the role of each input dataset. Our new framework to tackle the PMDC, based on GNN (Graph Neural Networks) is proposed and described in detail in Sec. IV. The experimental observations are reported in Sec. VI, and conclusions and directions for future work are summarized in Section VII.

II. Related Works

MDC and PMDC are relatively new concepts that have been recently proposed in [14] and, as a consequence, there are no other state-of-the-art methodologies that can be directly leveraged. Compared to our previous solution, which had only preliminary results using LSTM (Long Short-Term Memory), in this paper, we proposed a GNN [15] based framework inspired by travel-time estimation problem. In the rest of this section, we outline several related works in travel-time estimation, graph representation learning, HD maps, and MDC.

A. Travel-time estimation

Trip planning, ETA (Estimated Time of Arrival), as well as some related "derived" topics such as fuel consumption prediction [16] and electric vehicle energy management [17] are the closest ideas to our objective. Therein, the travel-time estimation – also known as ETA and Origin-Destination (OD) time estimation problem [18], [19] – is one of the widely used tasks in of high importance in location-based services/applications in both consumer market and industry. Given a pair of origin and designation locations (or the entire route), and prerequisite road network and other information/pattern (such as traffic, weather and accident), an accurate time predicting result not only benefits consumers' everyday life, but also the optimization of entire social system in the aspects such as logistic and ride-sharing [20].

The history of solving ETA problem can be traced back for decades, evolving from simple statistic model and regression [21] to modern convolutional neural network (CNN) based solutions [22], [23]. Numerous works using LSTM [24], [25], GNN [26]–[28], the hybrid of LSTM and GNN [29], and even image based [30] solutions have achieved impressive results. The concurrent works such as Curb-GAN [31] and DeepOD [32] integrate/embed external factors and historical data into the training process and show the strong correlation to prediction results.

B. LSTM and GNN

The fast development of LSTM has drawn substantial attention due to its ability to model the long-term/historical dependencies of time-series data such as speed, ETA [33], fuel/energy consumption [16], [17] for a single vehicle. Meanwhile, the lacking of information from adjacent "samples" (such as links/edges in a graph) limits the LSTM from learning surrounding knowledge.

Complementary, GNN-like architectures are potentially compatible with road networks and have inherent advantages over LSTM-like solutions. Conventional GNNs have disadvantages in handling the changes of the graph, not only the deletion/insertion of nodes, but also the ever-changing features of the nodes. Re-training the model is needed in order to represent this node [15], [34] once changes are obtained. Fortunately, being different from social network, maps and HD maps are relatively "stable"/staic on both topological and featurization aspects. Researchers working on dynamic graph representation learning [35], [36] are trying to eliminate such issues.

C. HD maps and Map Data Consumption

HD maps being used in AVs consist of at least lane boundary geometries, road signs, and other road furniture/objects,

composed by points (for lane boundaries) and voxels (for other furniture), attached with other descriptive tags and information. Due to extremely high level of detail, their data size is much larger than conventional maps. For references, in conventional maps, dozens of control points are sufficient to represent a hundred even a thousand meters long center line of a lane or road. In contrast, in HD maps, each urban lane meter can have over 1.2×10^4 and 4×10^3 voxels at resolutions of 10^{-1} meter and 2×10^{-1} meter respectively [6]. Voxels also have the advantage of potentially compatible with tree-like (i.e., quadtree and octree) hierarchy structures. There are mainly two types of containers that we can use to organize the road furniture which are distinct from the container dimensions: attach objects to one-dimensional road network and two-dimensional global tile [37]. One-dimensional data structure is rarely used [38], [39], because the objects do not have a global view, which may cause data redundancy (same furniture appears/obtained multiple times from different link/edge) and makes global optimization and alignment harder. Tile-like structure is the ideal container but inherently incompatible with road network (graph representation).

Similar to the definition of energy/fuel consumption in vehicle energy management study (i.e., miles per gallon or watt-hours per mile), MDC is based on the size of the maps data that a vehicle needs for its semi-autonomous or fully-autonomous driving function(s) [7]. One possibility to quantify the MDC is by the amount of data the vehicle needs to load for executing real-time applications. Specifically, the vehicle needs to load surrounding objects, represented in polygons/vertices or grids/voxels depending on the object representation. Even though representing objects in vectors can significantly reduce the size of the data and has invariance advantages such as scale and shift, raster representation is still more widely used in real-time autonomous driving applications since (cf. [7]): (1) sensors (e.g., LiDAR, depth/stereo cameras) equipped on vehicles acquire data that is either directly represented in raster format, or can straightforwardly converted into raster-like information. (2) most algorithms for autonomous (and assisted driving) applications, such as the ones used in vehicle self-localization, need to be fed with raster data [40].

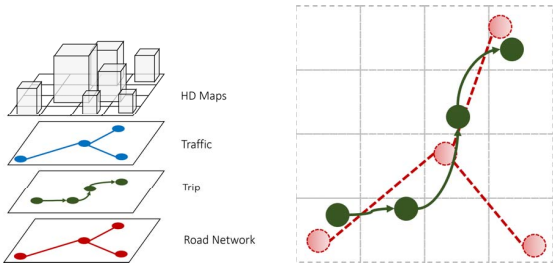


Fig. 1: The stacking of four types of data: HD maps, traffic, trip, and road network from top to bottom (left); the overlay of tiles (gray grids), a trip (green trajectory) and road network (red graph) (right).

If there are higher safety requirements (under certain internal and external factors/constraints), the resolution of retrieved voxels needs to be increased, causing a substantial increase in the size of the map data to be downloaded. The size of objects information that the vehicle needs is also highly dependent on the sensor configuration, such as refresh rate, layout and orientation, angular/spatial resolution, sensing range, and even vehicle's motion. On-board acquisition is irrelevant to our paper, we only focus on the data (size) retrieved from server.

What separates our work from the related literature is that we combine data from multiple heterogeneous but spatially correlated sources, as illustrated in Figure 1. More specifically, we embed the HD maps – or, in a broader sense, the tile/raster based data – as a part of the heterogeneous input of a graph representation learning architecture, and design a unique framework to train MDC problem with other important (e.g., traffic) data sources, and use it to solve the PMDC of a trip.

III. Preliminaries

We now describe the specifics of the data sources, and present formal definitions of the concepts used in the rest of this paper.

Maps and HD maps. We assume that an HD map is represented in the widely accepted 2D tile system for geo-regions, accompanied with a resolution value, denoted by $M \subseteq \mathbb{R}^{|P| \times |Q| \times |R|}$, where $|P|$ is the number of cells along x-coordinate; $|Q|$ is the number of cells along the y-coordinate of the suitably selected system; and R is the set of resolution values used among the cells. Typically, the values of R correspond to voxel-sizes (i.e., one can have different resolution levels for a given configuration of 2D tiles). The cell $(p, q, r) \in M$, denoted $m_{p,q,r}$, contains the voxels corresponding to single tile at tile coordinate $p \in P$, $q \in Q$ and resolution $r \in R$ in a respective grid of the geographical area of interest.

We assume a conventional road network represented as a directed graph $G = <V, E>$, where the elements of V (i.e., vertices) correspond to an intersection, and the elements of E (i.e., the edges) correspond to road segments. Each $v_i \in V$ has unique location, specified by its coordinates $(v_i.x, v_i.y)$. Similarly, each $e_k \in E$ is represented as the triplet $<u_k, v_k, w_k>$ where $u_k, v_k \in V$ are the start node and end node of e_k, while w_k denotes the "weight", which could stem from different context, such as: length, traffic travel index (TTI), or HD maps corresponding to e_k.

We further assume that M and G are specified in the same coordinate system. However, we note that an extra linear projection is still needed to calculate the conversion from G to M. To simplify the problem, we introduce $I_{(<p,q>,r)}$ to indicate the quantity of that data with respect to a tile location $<p, q>$ and resolution r, and $f_p(<x, y>) = <p, q>$ to indicate the project from geolocation $<x, y>$ to tile coordinate $<p, q>$. Given a vehicle at $<x, y>$ with a set of configuration c (a combination of internal and external factors, such as speed, acceleration, hardware configuration, weather and traffic), the vehicle needs the surrounding information not only consists of the current tile $f_p(<x, y>)$, but also nearby tiles, at certain resolution(s). We also define piecewise-defined

functions $f_d(c) = d$ and $f_r(c) = r$ which use c to determine a pair of tile search size $d, d \leq 0$ and resolution $r \in R$. Therefore, the HD maps information of vehicle at $< x, y, c >$ can be represented by a set B consists of HD maps indices, where

$$\begin{aligned} &B_{x,y,c} = \{f_p(<x, y>) + <p, q>, r\}, \\ &p \in [-f_d(c), f_d(c)], q \in [-f_d(c), f_d(c)], r = f_r(c) \end{aligned} \tag{1}$$

and then the MDC at single moment $< x, y, c >$ can be formalized as:

$$MDC_{x,y,c} = f(M, B_{x,y,c}) = \sum_{b \in B_{x,y,c}} I_b. \tag{2}$$

Trip. A trip, or a trajectory, is a sequence of geospatial points represented as $L = \{l_i\}$, where $l_i =< x_i, y_i, t_i, c_i >$ or $l_i =< x_i, y_i, z_i, t_i, c_i >$[1] is the i^{th} point of L. $<x_i, y_i>$, t_i and c_i denote the geo-location, timestamp and configuration respectively.

In this study, a raw trip L needs to be converted into a graph G representation and then processed with GNN framework. Let $L_m = <x'_i, y'_i, t_i, c_i>$ denote the map-matched L to graph G, and $L_g = <u_j, \tau_j, \epsilon_j>$ denote the trip in graph representation, where $<x'_i, y'_i>$ is the map-matched point of raw point $<x_i, y_i>$ at t_i, and $u_j \in V$ is a node between map-matched points $<x'_i, y'_i>$ and $<x'_{i+1}, y'_{i+1}>$, τ_j is the interpolated timestamp between t_i and t_{i+1}, and ϵ_j is the interpolated configuration of c_i and c_{i+1}, for the cases when $<x_i, y_i>$ and $<x_{i+1}, y_{i+1}>$ do not match to a same edge.

Define $dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>)$ is the in-graph distance between $<x_i, y_i>$ and $<x_{i+1}, y_{i+1}>$, in this example:

$$\begin{aligned} &dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>) \\ =&dist(<x'_i, y'_i>, <x'_{i+1}, y'_{i+1}>) \\ =&dist(<x_i, y_i>, u_j) + dist(u_j, <x_{i+1}, y_{i+1}>). \end{aligned} \tag{3}$$

Then we can define interpolated τ_j as

$$\tau_j = \frac{dist(u_j, <x_i, y_i>)}{dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>)} \times (t_{i+1} - t_i) + t_i. \tag{4}$$

as well as ϵ_j (assume c can be interpolated)

$$\epsilon_j = \frac{dist(u_j, <x_i, y_i>)}{dist(<x_i, y_i>, <x_{i+1}, y_{i+1}>)} \times (c_{i+1} - c_i) + c_i. \tag{5}$$

Note that if the trajectory sampling rate is sparse or many intersections are clustered together, multiple nodes may occur between two adjacent $<x'_i, y'_i>$ and $<x'_{i+1}, y'_{i+1}>$.

MDC. Assume there is a trip $L_t = \{<\hat{x}_k, \hat{y}_k, \hat{t}_k, \hat{c}_k>\}$ converted from a map-matched trip L_m, has a perfect sampling rate which let there is one and only one trajectory point locates in each adjacent tile, where $f_p(\hat{x}_k, \hat{y}_k) = f_p(\hat{x}_{k+1}, \hat{y}_{k+1}) \pm <\{0, 1\}, \{0, 1\}>$. To form such trajectory, if there is no such trajectory point locates in a tile from the map-matched trip L_m, a new point should be interpolated and the construction of its $\hat{c}$ follows Equation 4; if multiple trajectory points cluster in tile, the center trajectory point will be selected. Note that, L_m is unidirectional transferred from L, L_g and L_t are unidirectional transferred from L_m. $|L| = |L_m|$, but not necessary equals to $|L_g|$ or $|L_t|$.

Thus, the total MDC of trip L can be formalized as

$$\begin{aligned} &MDC_L = MDC_{L_t} = \\ &MDC_{<\hat{x}_1, \hat{y}_1, \hat{c}_1>} \uplus \cdots \uplus MDC_{<\hat{x}_{|L_t|}, \hat{y}_{|L_t|}, \hat{c}_{|L_t|}>}, \end{aligned} \tag{6}$$

where $\uplus$ denotes a special MDC accumulation operation which unions $\cup$ of two sets of HD tiles inside function f. For instance,

$$\begin{aligned} &MDC_{<\hat{x}_1, \hat{y}_1, \hat{c}_1>} \uplus MDC_{<\hat{x}_2, \hat{y}_2, \hat{c}_2>} = \\ &f(M, B_{\hat{x}_1, \hat{y}_1, \hat{c}_1}) \uplus f(M, B_{\hat{x}_2, \hat{y}_2, \hat{c}_2}) = \\ &f(M, B_{\hat{x}_1, \hat{y}_1, \hat{c}_1} \cup B_{\hat{x}_2, \hat{y}_2, \hat{c}_2}) = \sum_{b \in B_{\hat{x}_1, \hat{y}_1, \hat{c}_1} \cup B_{\hat{x}_2, \hat{y}_2, \hat{c}_2}} I_b. \end{aligned} \tag{7}$$

We note that, whenever there is no ambiguity, we will omit certain subscript(s) and/or superscripts. Thus, for example, to denote the MDC of a given trajectory L, we will use MDC_L when clear from the context.

IV. Methodology

In this section, we elaborate each module of the framework. As shown in Figure 2, our framework mainly consists of three components: **HD maps encoder**, which encodes the HD maps information of each edge into a fixed-length (d_m) vector; **Road segment encoder** converts each road segment into a d_r-length vector; **MDC generator** enabling the training for trip planning purposes.

Note a trip will be firstly map-matched and converted into an $|L_g|$-length vector in graph representation (cf. Section III). Traffic information, i.e., road conditions and traffic flow, has already been processed into a graph (for each edge at a specific timestamp, there is a d_t-dimensional vector), hence we do not further encode the traffic information.

A. MDC Generator

Since there is no real MDC data acquired by either experimental or consumer vehicles, we create an MDC for each trip following Equations 1, 2, 6 and 7 – using the trip, road network, HD maps and the related hard-coded variables as input. The process is illustrated in Figure 2. To reduce redundant description of this process, we elaborate the construction in Section V. The output of this module is a stand-along MDC value, which is denoted by MDC_L for trip L.

B. Road Segment Embedding

In general, the city road network consists of a set of interconnected road segments. Each of the segments denotes a sample of the physical connectivity. As for a specific trip, its map-matched trajectory can be split into a sequence of road segments sorted by time, and each segment is unique in the whole road network. Thus, road segment can be considered as the meta component of the trip and city road network. Since

[1]In this paper we only use 2D points/cells. Z-axis in coordinate system (e.g., altitude) should be considered in future 3D transportation, such as drone delivery.

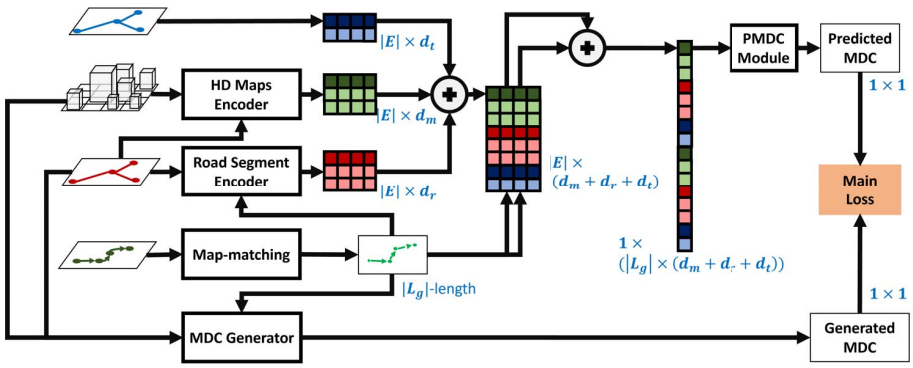


Fig. 2: The architecture of all the modules of PMDC solution. The dimensions are color-coded, shown next to each output.

the explicit knowledge about the underlying interaction has been extracted and saved in the topological graph structure, implicit representations of road segments are necessary for resolving trips. To enhance and ensure the conciseness, we have established the city road network according to historical trip trajectories in Section III. In this section, we introduce an approach for learning the road segment representation and preserving the similarity of neighboring segments in the embedding space.

Given the weighted road graph network $G =< V, E >$, we first use a $|V|$-dimensional one-hot vector $\mathbf{o}_i$ as the initial feature of node $v_i (v_i \in V)$, which attaches a unique representation to each of the nodes. However, one-hot representation can not fully reflect the connectivity of the city road network. For example, the standard similarity calculation – Euclidean distance between any two one-hot embeddings is the same. Spatially adjacent nodes, e.g., neighboring nodes, should be given close embeddings. Inspired by the similarity-preserving network representation methods [41], we leverage GraphWave [42], i.e., an unsupervised node embedding method, to extract the topological road structure and represent nodes' network neighborhood via a low-dimensional embedding. The process can be defined as,

$$U = \text{GraphWave}(G, X = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{|V|}]), \tag{8}$$

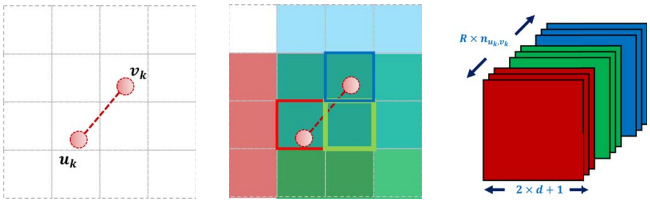
where $X = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{|V|}]$ is the input one-hot embedding matrix, G denotes the topological structure of the road network and U represents the learned node embedding vectors.

However, in this study, road segments, i.e., edges E of the graph G, are the key components of each trip that we want to represent. For example, if $e_k \in E$ and $e_k =< u_k, v_k, w_k >$ is a part of a trip, the car will first pass through u_k. Thus, to take the direction of road segments into consideration, we splice two nodes' representation by order with learn node embedding and leverage the concatenation to determine the spatial feature of edge. Specifically, for a given edge e_k, its hidden representation can be defined as,

$$\mathbf{e}_k = \mathbf{u}_k \oplus \mathbf{v}_k, \tag{9}$$

where the the $\oplus$ symbol refers to the concatenation of embeddings. We note that the edge weight w_k has been considered in the node embedding process, i.e., the road network G is weighted, thus we omit it here.

C. HD maps Embedding



(a) (b) (c)

Fig. 3: Illustration of HD maps information featurization: (a) one road segment $e_k = u_x \rightarrow v_x$, (b) three sequential tiles $red \rightarrow green \rightarrow blue$ belong to e_k, and (c) HD maps information $\mathbf{HD}_{e_k}$ assigned to this edge.

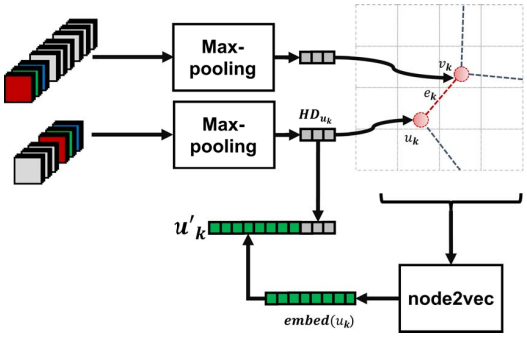


Fig. 4: Architecture of HD maps encoder.

Recall (cf. Section III) that at tile $< p, q >$, with given search window size d (tiles), the surrounding HD information can be represented as $[M_{p-d \rightarrow p+d, q-d \rightarrow q+d, 1 \rightarrow R}]$, which is a 3-dimensional vector. Assume an edge $e_k \in G$ with start and end points $u_k, v_k \in V$ (shown in Figure 3 (a)) has a n_{u_k,v_k}-length sequential footprint $\{< p_{u_k}, q_{u_k} >, \ldots, < p_{v_k}, q_{v_k} >\}$ in HD maps tile coordinate, where $< p_{u_k}, q_{u_k} >= f_p(u_k.x, u_k.y)$ (illustrated in red $\rightarrow$ green $\rightarrow$ blue small boxes in Figure 3 (b)). Thus, the HD maps $\mathbf{HD}_{e_k}$ information of

edge e_k can be represented as a 4-dimensional vector

$$\begin{aligned}&\mathbf{HD}_{e_k} = \\ &[[M_{p_{u_k}-d\to p_{u_k}+d, q_{u_k}-d\to q_{u_k}+d, 1\to R}], \ldots, \\ &[M_{p_{v_k}-d\to p_{v_k}+d, q_{v_k}-d\to q_{v_k}+d, 1\to R}]], \\ &\mathbf{HD}_{e_k} \in \mathbb{R}^{(2d+1)\times(2d+1)\times R\times n_{u_k,v_k}}\end{aligned} \tag{10}$$

Where n_{u_k,v_k} denotes the distance from u_k to v_k in tile coordinate.

The purpose is to extract a fix-sized feature vector for each edge using the associated HD maps features and the graph connections. We utilize the nodes and edges in all trips to build a graph neural network and apply max-pooling to the last dimension (across its batches) of the HD maps associated with a node to obtain the HD maps features for each node $\mathbf{HD}_{u_k}$. To be specific, given a set of edges $\{e_j\}$ connect to node u_k, $\mathbf{HD}_{u_k} = \mathbf{Maxpool}([\mathbf{HD}_{e_j}])$.

Inspired by the *word2vec* [43] work in natural language processing studies, we create a sliding window among nodes in our trips and maximize the probabilities of two connected nodes being on the same trip. We first generate node embeddings $embed(u_k)$ using skipgram and negative samplings in node2vec [44]; we then concatenate the node embeddings and the pooled HD maps features $\mathbf{u}' = [embed(u_k), \mathbf{HD}_{u_k}]$ via Hadamard transform. Similar to the process used in the previous section (cf. Equation 9), the final HD maps embeddings $\mathbf{e}'_k$ for edge e_k can calculated as $\mathbf{e}'_k = \mathbf{u}'_k \oplus \mathbf{v}'_k$ with a dimension of d_m. The architecture of HD maps encoder is illustrated in Figure 4.

D. PMDC

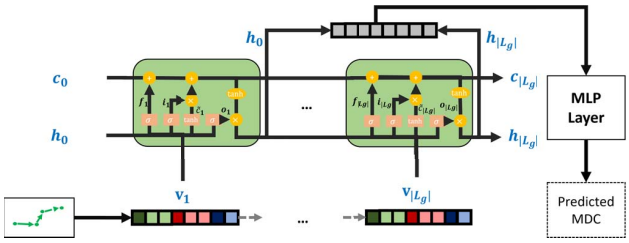


Fig. 5: Architecture of PMDC module.

Aforementioned, a trip L has an equivalent representation in graph which is $L_g = \{<u_j, \tau_j, \epsilon_j>\}$, and this representation can be further converted in to a sequential sets of adjacent edges which denoted as $P = <<u_1, u_2>, \ldots, <u_{J-1}, u_J> = <e_1, e_2, \ldots e_{J-1}>>$, where $J = |L_g| - 1$. The outputs of **Road Network Encoder** and **HD Maps Encoder** are denoted as $\mathbf{e}$ and $\mathbf{e}'$, respectively, and the road network traffic information directly pulled from the dataset is denoted by $\mathbf{t}$. For each $e_i \in P$, we construct a fixed-length vector $\mathbf{v_i}$ by concatenating each embedding of this edge to get the final embedding $\mathbf{v_i} = [\mathbf{e}_i, \mathbf{e}'_i, \mathbf{t}_i]$.

Given its effectiveness in summarizing the contextual information from sequential data, we utilize LSTM to encode the trajectory knowledge into a fixed-length vector from historical segments, and each road segment e_j of a trip L_g is an LSTM time step $LSTM(i)$ defined by

$$\begin{aligned}e_j &= \sigma(W_e[h_{j-1}, \mathbf{v}_j] + d_e), \\ f_j &= \sigma(W_f[h_{j-1}, \mathbf{v}_j] + d_f), \\ o_j &= \sigma(W_o[h_{j-1}, \mathbf{v}_j] + d_o), \\ \tilde{c_j} &= tanh(W_c[h_{j-1}, \mathbf{v}_j] + d_c), \\ c_j &= f_j \otimes c_{j-1} + e_j \otimes \tilde{c_j}, \\ h_j &= o_j \otimes tanh(c_j).\end{aligned} \tag{11}$$

The input, forget, and output gates are e_j, f_j, and o_j, respectively, which represent how much information we extract from the current input, save from the previous hidden state, and keep in current output. The hidden state h_j indicates the sequential embeddings, and c_j represents the contextual embeddings. $\tilde{c_j}$ denotes the intermediate embeddings carried out from input contexts. Weight matrices W_e, W_f, W_o, W_c and bias vectors d_e, d_f, d_o, d_c are shared across different trips. The initial hidden vector h_0 is a vector of zeros.

$$\widehat{MDC} = W_f h_{J+1} + b_f. \tag{12}$$

We utilize the hidden state of the final step h_{J+1} to embed one complete trip and append one multilayer perceptron (MLP) layer to obtain the predicted MDC value $\widehat{MDC}$; we use the standard mean squared error (MSE) as the loss function. Equation 12 shows the prediction function, where W_f and b_f are trainable weights for converting dimension $|h_{J+1}|$ to 1. The structure of PMDC module is illustrated in Figure 5.

V. Data Preparation

Other than the deep learning framework, the data is another pillar for tackling any learning problem. As mentioned, since MDC and PMDC are new concepts, there is no existing dataset collected or built for this specific task. We integrate heterogeneous data from various sources to create what we call Synthetic City Dataset (SCD), which consists of road network (in graph representation) from OSM [45], traffic information [46] (in plain text) and trips from DiDi Open Dataset [46] and HD maps model from previous work [7].

In this paper, we make several important improvements on the SCD with respect to [14]. Below is a concise summary of the SCD:

- **Road network**, an $8,060$ meters (longitudinal) by $8,053$ meters (lateral) area of Xi'An, consists of $1,814$ OSM links (roads) and $7,421$ OSM nodes. A graph with $4,771$ edges and $2,140$ nodes is generated when converting the raw OSM data into graph representations.
- **HD map**, a map created by attaching the voxel/occupancy-grid distribution (at different resolutions) learned from Chicago to Xi'An road network randomly (with several hard-coded constraints).
- **Traffic**, a "week calendar" size – which has a size of 7 days by 6×24 time sections (TTI information is recorded every 10 minutes for each link) per day – matrix contains

traffic information of each link. 53.20% of edges have traffic information recorded, and accounts 61.82% of the total length of the road network. If there is no traffic information attached to an edge, 0s will be assigned.

- **Trips**, $113,976$ filtered and processed trips from the DiDi trip dataset which contains over 3 millions trips.

Note, due to the data security policy, The data (both trips and traffic) provided by DiDi is enforced to be encoded in *GCJ-02* coordinate [47]. Which, in turn, a consistent (locally) misalignment[2] between DiDi's coordinate and OSM coordinate is obtained (by manually aligning several intersections).

Map-matching When switching from LSTM-based solution [14] to a GNN-based architecture, a serious map-matching is prerequisite. After testing out several solutions, we select Fast Map-Matching (FMM) [48], [49] – a hidden Markov model based solution with pre-computation of an upper bounded origin-destination hash table for acceleration purpose – due to its high accuracy, speed and accessibility. The configurations of FMM are 8 nearest neighbours (edges), $3 \times 10^{-3\circ}$ search radius (approximately 300 meters) and a GPS error of $5 \times 10^{-4\circ}$ (approximately 50 meters).

Trip MDC When generating the MDC for a given trip, some rules/configurations need to be "hard-coded", such as search range d and resolution $r \in R$ at different vehicle internal parameters (motion), along with values of external factors at a given time instant. Ideally, the sensors (and the hardware) are inclined to keep a high/consistent acquisition rate and quality to ensure driving safety. Unfortunately, due to hardware limitations, a trade-off between sampling rate (maintaining sampling rate and lower the data quality/resolution) and data quality (keep data resolution and drop frames) rises [50], [51]. Most solutions tend to be the first solution to fulfill AV's reaction time requirement [52].

In DiDi's trip dataset, the only motion information recorded is the velocity of each trajectory point. Thus, for each trajectory point's velocity c, we define two thresholds $\Gamma_1 = 5m/s$ and $\Gamma_2 = 10m/s$ to determine different d, r combinations into three segments as searching criteria [3]. $d_{1,2,3} = 1, 3, 5$ and $r_{1,2,3} = 13, 12, 11$ [4].

VI. Experimental Results

In this section, we compare our result to several baseline approaches being used in similar tasks and discuss the impacts/effectiveness of each encoder in our framework. The full length of embedding is $d_v = |\mathbf{v}| = |[\mathbf{e}, \mathbf{e}', \mathbf{t}]|$, where each component has a length of $d_m = 100$, $d_r = 128$ and $d_t = 2$. We note that, for reproducibility, the SCD data and the source codes are publicly available at https://github.com/zangandi/HDMapsDataPrediction.

[2]The misalignment from OSM coordinate to DiDi coordinate is $[-0.0016°, 0.0047°]$ roughly equivalents to a 468.1 meters ground resolution.

[3]The reason we use these two thresholds is the distribution of vehicle speed (at certain time interval) learned from the dataset follows normal distribution with $\mu \approx 8m/s$.

[4]Tile level 13 equivalents to a $\approx 10^{-1}$ meter voxel size, $12 \longrightarrow 2 \times 10^{-1}$, and so forth.

A. Baseline

Due to the novelty of the PMDC, there are no approaches that we are aware of that can be categorized as related ones. Hence, for complementary perspectives, we use the following approaches as baselines:

a) ***LSTM****:* ([14]) an LSTM based solution that concatenates all the sequential HD maps tile information along a given trip, combined with related internal and external features such as velocity and traffic. This pipeline is extremely expensive (both computation-wise and storage-wise) when encodes each trip since the HD maps are represented in their raw format. No neighbor (global) information is encoded.

b) ***Linear Regression****:* ([53]) a Linear Regression (LR) model is trained to minimize the loss (Euclidean distance) between predict MDC and true MDC to solve the PMDC. The complexity of building such feature vector (and normalized to a fixed-length vector) is the same as it in the LSTM pipeline. One of the significant drawbacks of this type of solution is the lack of representing both sequential spatial and temporal information.

c) ***DeepOD****:* ([32]) Deep Origin-Destination is a neural network based solution that learns and encodes both spatial and temporal properties from adjacent edges and all given trips to represent the current edge. The final travel estimation model is trained by concatenating each sequential edge from different trips and a Multilayer perceptron (MLP). Note this work aims to solve the ETA problem and is irrelevant to our case. Hence, to make a fair comparison, we add the HD maps information into the training process as a part of the embedding.

B. Evaluation Metrics

To systematically evaluate the performance of PMDC, popular assessment criteria such as Mean Absolute Error (MAE), Mean Absolute Percent Error (MAPE) are used in our paper. Let MDC_L and $\widehat{MDC}_L$ denote the ground truth MDC and predicted MDC of a trip $L \in$ dataset S, the MAE and MAPE of the entire set can be computed as,

$$\begin{aligned} \textit{MAE}_S &= \frac{1}{|S|} \sum_{L \in S} |MDC_L - \widehat{MDC}_L|, \\ \textit{MAPE}_S &= \frac{1}{|S|} \sum_{L \in S} \frac{|MDC_L - \widehat{MDC}_L|}{MDC_L}. \end{aligned} \tag{13}$$

To reflect the performances on the entire dataset consisting of the varying length trips, we also bring in weighted MAPE (wMAPE) in the evaluation, where the weight is the travel distance of the trip:

$$\textit{wMAPE}_S = \sum_{L \in S} \frac{\textit{length of } L}{\textit{total length of } S} \frac{|MDC_L - \widehat{MDC}_L|}{MDC_L} \tag{14}$$

C. Comparison with Baselines

Firstly, the MAEs, MAPEs and wMAPEs of the experiment results are shown in Table I, and the Probability Density Function (PDF) of MAPEs is illustrated in Figure 6 (top).

	MAE ("voxels")	MAPE %	wMAPE %
LR	$7.67e^6$	21.58	22.81
LSTM	$1.82e^7$	53.66	—-
DeepOD	$4.54e^6$	15.25	15.30
Ours	$3.41e^6$	11.08	11.19

TABLE I: Experiment results on test set with different models. Note experiment LSTM has no wMAPE value assigned because in that experiment, all trip lengths are normalized.

The first thing to catch our sight is the extremely poor performance of LSTM. Aforementioned, this LSTM framework is borrowed from our previous work, which is designed and optimized for the objective of giving a sequence of embeddings from previous trajectory points within a time window, predicting the MDC for the next time interval. We modify this work by simply expanding the size of time window to the entire trip for predicting the MDC.

Secondly, the LR's performance is impressive. Recall previous sections, the HD maps – or to be precise, I, the number of objects of an HD map tile – are generated using the normal distribution learned from real-world data and applied/attached to Xi'an's road network randomly. In real-world scenario, the distribution of "objects" is not only spatial-wise, but also graph-wise unique, varies from districts/blocks functionalities. For instance, central business districts consist of a higher volume of objects than park districts. Once the voxel distribution is applied to the entire city, the discrimination of number of voxels is eliminated. At the same time, the generation of trip MDC follows a straightforward piecewise-defined function with several hard-coded variables, no regularization applied.

Even though the synthesizing of HD maps and trip MDCs have such drawbacks, the performance of our proposed model is ahead of LR and the DeepOD. Considering the low margin of performance difference, a 4.17% improvement of MAPE shows the effectiveness of the integration of HD maps encoder and road network traffic information. Once real-world data is used, a dilated performance margin is expected.

The introduction of wMAPE shows whichever module is deployed, the trips with longer travel distance intending to have a worse prediction result in MDC.

D. Effectiveness of Embeddings

Since there are multiple modules integrated in our framework, we disable different modules to evaluate the effectiveness of each embedding. As aforementioned, we have three embeddings: road network, HD maps, and traffic information, which leads to 2^3 experiments. Note, if ether road network embedding or HD maps embedding is disabled, we will use one-hot vector to replace the graph embedding, and the experiments on some embedding combinations are meaningless, which we will ignore. Given the fixed order of "road network, HD maps, traffic", we use a 3-digit abbreviation – $\{111, 110, 101, 100, 010, 011\}$ – to denote each experiment combination. The MAPEs of experiments are shown in Table II

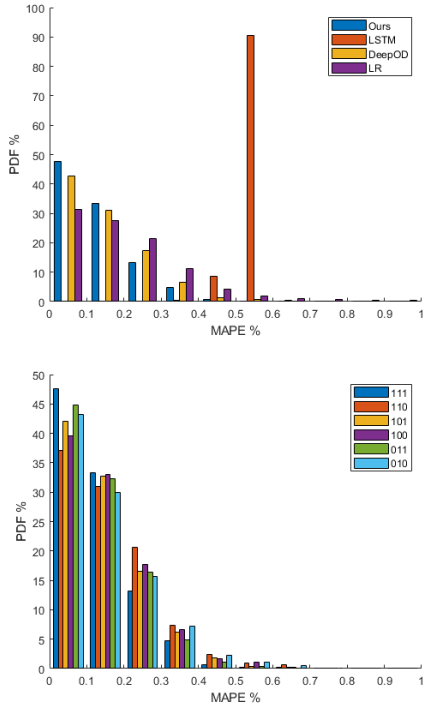


Fig. 6: Probability Density Functions (PDFs) of MAPES on the test set using different methodologies (top), and the effectiveness experiments (bottom).

and PDFs are shown in Figure 6 (bottom), from which the following observations are made:

1) Comparing all the controlled trials of traffic information (i.e., $\{111 \leftrightarrow 110\}$, $\{101 \leftrightarrow 100\}$ and $\{011 \leftrightarrow 010\}$), the integration of such information significantly improves the performance with the combination of HD maps encoding ($11.08\% = 13.34\% - 2.26\%$ and $11.58\% = 14.38\% - 2.8\%$). Once the traffic information stands alone, less improvement can be obtained ($14.22\% = 14.43\% - 0.21\%$);
2) Both HD maps and road network encoders do not contribute the performance as well as the traffic information when being concatenated independently (i.e., $\{110 \leftrightarrow 100\}$ and $\{110 \leftrightarrow 010\}$). The performances only rise by 1.09% and 1.04% respectively;
3) The higher wMAPEs (over MAPEs) and Figure 7 (left) both indicate the majority of longer (distance-wise) trips have worse prediction results than shorter trips. The common issue where a certain amount of "outliers" appear on the shorter-trip-end can also be found in similar trip property estimation problems [14], [54], which is caused by the amplification of features' volatility in a shorter sampling time. The quantified performance drops are $0.11\%, 0.16\%, 0.20\%, 0.35\%, 0.48\%, -0.07\%$ for six experiments respectively, and have positive correlations with their overall performances;
4) When switch the weights from trip length to trip duration (i.e., $\frac{duration\ of\ L}{total\ duration\ of\ S}$), another observation (from Fig-

code	MAPE % (change %)	code	MAPE % (change %))
111	11.08 (0.00)	***100***	14.43 (3.35)
110	13.34 (2.26)	***010***	14.38 (3.30)
101	14.22 (3.14)	***011***	11.58 (0.50)
code	**wMAPE^1 % (change %)**	**code**	**wMAPE^1 % (change %))**
111	11.19 (0.00)	***100***	14.78 (3.59)
110	13.50 (2.31)	***010***	14.86 (3.67)
101	14.42 (3.23)	***011***	11.51 (0.32)
code	**wMAPE^2 % (change %)**	**code**	**wMAPE^2 % (change %))**
111	11.35 (0.00)	***100***	14.35 (3.00)
110	13.76 (2.41)	***010***	14.56 (3.21)
101	14.18 (2.83)	***011***	11.81 (0.46)

TABLE II: MAPE and wMAPEs of effectiveness experiments. $\mathbf{wMAPE^1}$ and $\mathbf{wMAPE^2}$ denote the wMAPE use trip distance and duration as weights.

ure 7 (right) and Table II) is the performance difference of each experiment has a negative correlation to its trip duration. The performance drops are, respectively: $0.27\%, 0.42\%, -0.04\%, -0.08\%, 0.18\%, 0.23\%$ respectively. This result shows a better embedding (with temporal information integrated) is more effected by extending the trip duration.

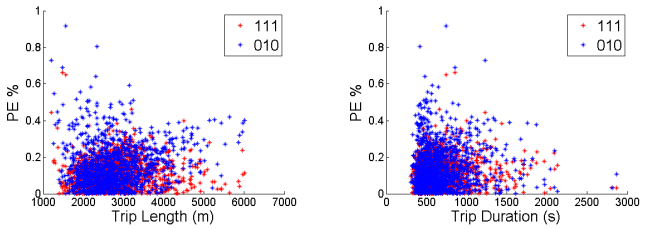


Fig. 7: Distribution of Percentage Error (PE) to trip length (left) and trip duration (right) of experiments 111 (best performance) and 010 (worst case).

VII. Concluding Remarks and Future Directions

In this paper, we studied the PMDC problem – a novel challenge for AVs, which aims to predict the HD maps data consumption with given HD maps, road network, and predetermined route. In comparison with its recent introduction [14], we have revised the definition of MDC and we have also improved the dataset based on integrating heterogeneous information sources. We proposed a comprehensive deep learning approach and neural network architecture, which is able to not only exploit historical trips, but also encode the HD maps of each edge into a fixed-length embedding and trained with a graph-based neural network instead of a naive one-hot encoding or average pooling. As our experiments have demonstrated, our model achieved superior performance over the conventional LR, LSTM and modified GNN based solutions (designed for similar objectives). A detailed study on module effectiveness not only shows the contribution of each encoder, but also indicates the robustness/consistency to trip length and duration. A future model can tweak the embeddings based on its objective (e.g., prediction preferences).

Since, to our knowledge, this is a forerunner work for the MDC problem, we recognize certain, at this point unavoidable, limitations. One notable limitation is the lack of real-world data. Both SCD (especially HD maps) and the MDC of each trip are constructed by following straightforward algorithms. Specifically, to generate the MDC, vehicle speed is the only input/factor that affects two variables: search window size and resolution. We believe that numerous other factors (both internal and external) from multiple software and hardware component, would create a more sophisticated variant of the MDC problem and enable more realistic settings to be tackled. One feasible solution in the short run is generating the map data and acquiring real-time vehicle configurations from 3D AV simulation platforms.

A specific variant of the problem that we want to address in the future is the one which would enable incorporation of the impact of (partial) HD maps updates. Such cases occur in the settings in which a new, possibly long-lasting, construction project has started, affecting larger metropolitan area. We are planning to investigate their efficient propagation into the corresponding embedding.

Acknowledgements

Research supported in part by the National Science Foundation SWIFT Grant No. 2030249, and the National Natural Science Foundation of China under Grant No.62176043 and No.62072077.

References

[1] K. Kuru and W. Khan, "A framework for the synergistic integration of fully autonomous ground vehicles with smart city," *IEEE Access*, vol. 9, pp. 923–948, 2020.

[2] H. G. Seif and X. Hu, "Autonomous driving in the icity—hd maps as a key challenge of the automotive industry," *Engineering*, vol. 2, no. 2, 2016.

[3] J. Neil, L. Cosart, and G. Zampetti, "Precise timing for vehicle navigation in the smart city: an overview," *IEEE Communications Magazine*, vol. 58, no. 4, pp. 54–59, 2020.

[4] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *CoRR*, vol. abs/1610.03295, 2016.

[5] K. Jo, C. Kim, and M. Sunwoo, "Simultaneous localization and map change update for the high definition map-based autonomous driving car," *Sensors*, vol. 18, no. 9, 2018.

[6] A. Zang, Z. Li, D. Doria, and G. Trajcevski, "Accurate vehicle self-localization in high definition map dataset," in *1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles*. ACM, 2017, p. 2.

[7] A. Zang, S. Luo, X. Chen, and G. Trajcevski, "Real-time applications using high resolution 3d objects in high definition maps (systems paper)," in *ACM SIGSPATIAL*, 2019, pp. 229–238.

[8] S. of California DMV, "DISENGAGEMENT REPORTS," https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/, 2020, online.

[9] J. Laborda and M. J. Moral, "Automotive aftermarket forecast in a changing world: The stakeholders' perceptions boost!" *Sustainability*, vol. 12, no. 18, p. 7817, 2020.

[10] S. Trommer, L. Kröger, and T. Kuhnimhof, "Potential fleet size of private autonomous vehicles in germany and the us," in *Road Vehicle Automation 4*. Springer, 2018, pp. 247–256.

[11] K. Jiang, D. Yang, C. Liu, T. Zhang, and Z. Xiao, "A flexible multi-layer map model designed for lane-level route planning in autonomous vehicles," *Engineering*, vol. 5, no. 2, pp. 305–318, 2019.

[12] X. Chen, "Hd live maps for automated driving: an ai approach," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2018, pp. 1–1.

[13] N. D. Standard, "Navigation data standard - open lane model documentation," Navigation Data Standard, Tech. Rep., 2016.

[14] A. Zang, X. Zhu, Y. Guo, F. Zhou, and G. Trajcevski, "Towards predicting vehicular data consumption," in *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2021, pp. 109–114.

[15] J. Skardinga, B. Gabrys, and K. Musial, "Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey," *IEEE Access*, 2021.

[16] S. Wickramanayake and H. D. Bandara, "Fuel consumption prediction of fleet vehicles using machine learning: A comparative study," in *2016 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 2016, pp. 90–95.

[17] T. D. Gaikwad, Z. D. Asher, K. Liu, M. Huang, and I. Kolmanovsky, "Vehicle velocity prediction and energy management strategy part 2: Integration of machine learning vehicle velocity prediction with optimal energy management to improve fuel economy," SAE Technical Paper, Tech. Rep., 2019.

[18] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 858–866.

[19] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire *et al.*, "Eta prediction with graph neural networks in google maps," *arXiv preprint arXiv:2108.11482*, 2021.

[20] D. Bertsimas, A. Delarue, P. Jaillet, and S. Martin, "Travel time estimation in the age of big data," *Operations Research*, vol. 67, no. 2, pp. 498–515, 2019.

[21] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: a survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009.

[22] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[24] Y. Duan, L. Yisheng, and F.-Y. Wang, "Travel time prediction with lstm neural network," in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE, 2016, pp. 1053–1058.

[25] N. C. Petersen, F. Rodrigues, and F. C. Pereira, "Multi-output bus travel time prediction with convolutional lstm neural network," *Expert Systems with Applications*, vol. 120, pp. 426–435, 2019.

[26] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 25–34.

[27] R. Li, G. Rose, and M. Sarvi, "Evaluation of speed-based travel time estimation models," *Journal of transportation engineering*, vol. 132, no. 7, pp. 540–547, 2006.

[28] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1695–1704.

[29] Z. Lu, W. Lv, Z. Xie, B. Du, and R. Huang, "Leveraging graph neural network with lstm for traffic speed prediction," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 74–81.

[30] T.-y. Fu and W.-C. Lee, "Deepist: Deep image-based spatio-temporal network for travel time estimation," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 69–78.

[31] Y. Zhang, Y. Li, X. Zhou, X. Kong, and J. Luo, "Curb-gan: Conditional urban traffic estimation through spatio-temporal generative adversarial networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 842–852.

[32] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2135–2149.

[33] Z. Lu, W. Lv, Y. Cao, Z. Xie, H. Peng, and B. Du, "Lstm variants meet graph neural networks for road speed prediction," *Neurocomputing*, vol. 400, pp. 34–45, 2020.

[34] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[35] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[36] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[37] K. Wong, Y. Gu, and S. Kamijo, "Mapping for autonomous driving: Opportunities and challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 91–106, 2020.

[38] TomTom, "Tomtom hd map with roaddna," 2017. [Online]. Available: https://automotive.tomtom.com/automotive-solutions/automated-driving/hd-map-roaddna/

[39] L. Li, M. Yang, C. Wang, and B. Wang, "Road dna based localization for autonomous vehicles," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, 2016, pp. 883–888.

[40] Y. Xu, V. John, S. Mita, H. Tehrani, K. Ishimaru, and S. Nishino, "3d point cloud map based vehicle localization using stereo camera," in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 487–492.

[41] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, 2019.

[42] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *SIGKDD*, 2018, pp. 1320–1329.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[44] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[45] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[46] DiDi, "DiDi Chuxing GAIA Open Dataset," https://gaia.didichuxing.com, 2019.

[47] P. N. W. Site, "Surveying and mapping law of the people's republic of china [eb/ol]."

[48] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2018. [Online]. Available: https://doi.org/10.1080/13658816.2017.1400548

[49] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.

[50] X. Xu, X. Wang, X. Wu, O. Hassanin, and C. Chai, "Calibration and evaluation of the responsibility-sensitive safety model of autonomous car-following maneuvers using naturalistic driving study data," *Transportation research part C: emerging technologies*, vol. 123, p. 102988, 2021.

[51] C. Basu, Q. Yang, D. Hungerman, M. Sinahal, and A. D. Draqan, "Do you want your autonomous car to drive like you?" in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 2017, pp. 417–425.

[52] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: disengagements, accidents and reaction times," *PLoS one*, vol. 11, no. 12, p. e0168054, 2016.

[53] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 276–281, 2004.

[54] K. Liu, Z. Asher, X. Gong, M. Huang, and I. Kolmanovsky, "Vehicle velocity prediction and energy management strategy part 1: Deterministic and stochastic vehicle velocity prediction using machine learning," SAE Technical Paper, Tech. Rep., 2019.