Path Travel Time Estimation using Attribute-related Hybrid Trajectories Network

Xi Lin
xlin012@e.ntu.edu.sg
Nanyang Technological University

Yequan Wang*
tshwangyequan@gmail.com
Advertising and Marketing Services, Tencent Inc.

Xiaokui Xiao
xkxiao@nus.edu.sg
National University of Singapore

Zengxiang Li
liz@ihpc.a-star.edu.sg
Institute of High Performance Computing, Singapore

Sourav S. Bhowmick
assourav@ntu.edu.sg
Nanyang Technological University

ABSTRACT

Estimation of path travel time provides great value to applications like bus line designs and route plannings. Existing approaches are mainly based on single-source trajectory datasets that are usually large in size to ensure a satisfactory performance. This leads to two limitations: 1) Large-scale data may not always be attainable, e.g. city-scale public bus data is usually small compared to taxi data due to relative fewer bus trips in a day. 2) Considering only single-source trajectory data neglects the potential estimation-improving insights of external data, e.g. trajectory dataset of other vehicle sources obtained from the same geographical region. A challenge is how to effectively utilize such other trajectory sources. Moreover, existing work does not attend the important attributes of a trajectory including vehicle ID, day of week, rainfall level etc., which are important for estimating the path travel time. Motivated by these and the recent successes of neural network models, we propose Attribute-related Hybrid Trajectories Network (AtHy-TNet), a neural model that effectively utilizes the attribute correlations, as well as the spatial and temporal relationships across hybrid trajectory data. We apply this to a novel problem of estimating path travel time of a type of vehicles using a hybrid trajectory dataset that includes trajectories from other vehicle types. We demonstrate in our experiments the benefits of considering hybrid data for travel time estimation, and show that AtHy-TNet significantly outperforms state-of-the-art methods on real-world trajectory datasets.

KEYWORDS

deep learning; path travel time; hybrid trajectory

*This work was done when Yequan was a Ph.D student at Tsinghua University and a visiting Ph.D student at Nanyang Technological University.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357927

ACM Reference Format:
Xi Lin, Yequan Wang, Xiaokui Xiao, Zengxiang Li, and Sourav S. Bhowmick. 2019. Path Travel Time Estimation using Attribute-related Hybrid Trajectories Network. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357927

1 INTRODUCTION

The knowledge of the travel time of any path within a road network is valuable to many. Through the lens of public transportation, this information is useful for bus line designs and road traffic monitoring. From the perspectives of ride-hailing (e.g. Didi, Grab and Uber) and courier delivery businesses, it improves user experience, and enables cost (e.g. time and fuel) savings, respectively. Such information also helps commuters in their trip-plannings.

With the growth in data collection technology, massive data is increasingly available for use, and to date, there has been much work which focuses on path travel time estimation using trajectory data. Estimating path travel times has been challenging due to multi-nature considerations. First, travel time is affected by spatial factors, such as the road characteristics. For example, a vehicle may travel faster along an expressway than within the residential district. Second, it is also temporally dependent. For example, travel times in the downtown area may be longer during peak hours due to traffic congestions. Other factors that affect such estimations are such as the vehicle type and weather conditions. For example, taxi sedans tend to move faster than large public buses.

Existing work traditionally falls under two approaches. The first is link-based methods, where the intuition is to first estimate the travel time of every road-segment, before combining them to get the path travel time [12, 19, 25, 28, 31]. These methods have a limitation in that they do not consider traffic complexities outside the road-segments, such as traffic lights. The second class of methods resolves such issues to some extent; the main idea is that travel time can be estimated by extracting the historical travel time of its sub-paths, which implicitly contains certain travel costs unconsidered in the link-based ones [5, 16, 30, 32]. More recently, deep learning has demonstrated further success for this task, based on its ability for powerful representation and implicitly modeling multi-natured traffic complexities [26, 33].

However, existing work mostly leverages single-source trajectory data that are usually also large in size. While having large-scale

data benefits estimation since it usually also means larger road coverage, data of such size is not always attainable. For example, public bus trajectory data sizes in many cities of the world are small due to the much fewer number of daily trips and the less extensive coverage of the road network, as compared to other types of vehicles like taxis. A newcomer to ride-sharing in a city also does not have access to large data due to the low number of trips that it has serviced initially. For the same reasons, the lack of large-scale data exists for vehicles where accurate path travel time estimation is valuable for highly-critical timely arrivals, e.g. ambulances. It is also prevalent for vehicles that require such estimation for global optimizations to meet business goals, e.g. logistics trucks.

Moreover, considering trajectory data from single sources neglects the potential estimation-improving insights present in other sources. To the best of our knowledge, there has yet to exist any work that leverages hybrid trajectory dataset for travel time prediction based on deep learning, e.g. using a mixture of bus trajectory data and a massive taxi trajectory data within the same city for bus travel time estimation. Estimating across hybrid trajectories, however, is also a challenging task mainly due to differences in trajectory characteristics. For example, buses tend to move slower than taxis due to their larger sizes and passenger loads, and have to stop at scheduled bus stops for pick-ups and drop-offs. As a result, attributes that come along with these trajectories tend to be highly crucial for such estimations, with some relatively more important than others, e.g. vehicle ID that reveals the vehicle type. However, existing work generally does not consider the relationships between attributes, nor attend the important ones.

As such, we propose Attribute-related Hybrid Trajectories Network (AtHy-TNet), the first end-to-end neural architecture for travel time estimation using hybrid trajectory data, to tackle the issues above. Its value lies in its applicability for use-cases when only small training data is available to the transport operator, e.g. public buses, shuttle buses, and urban logistics trucks. It also enables small/medium transport operators to enhance their services by leveraging massive trajectory data which are owned by giant operators or government agencies. To this end, we have considered various spatio-temporal and attributional treatments. The main contributions of this paper are summarized as follows:

- For the first time, we propose a neural method that effectively tackles the novel problem of estimating path travel time using hybrid data sources. More importantly, AtHy-TNet implicitly models the multi-nature factors affecting path travel time, and their correlations across trajectories of heterogeneous vehicle sources.
- We propose: i) an Attribute Correlation Module that models path attributes correlations and attend to the more important attributes, ii) a Sub-path Representation Module that utilizes a Tri-CNN Encoder and on-road information to create a representation for each sub-path, iii) a Sub-path Correlation Module that models correlations across sub-paths, and utilizes a Correlation component to learn the path-attribute relationships, and iv) a Joint-Learning Prediction Module that learns the path travel time estimation task with an auxiliary task of sub-path travel time estimation.
- We conduct extensive experiments on trajectory data obtained in Singapore. We demonstrate the benefits of considering hybrid data for travel time estimation, and show that AtHy-TNet significantly outperforms state-of-the-art methods. We also show the robustness of its performance across different conditions, e.g. different days of the week.

2 PRELIMINARIES

A trajectory of a vehicle within a road network is defined as a sequence of GPS records, $T_x = (x_1, x_2, x_3 ..., x_n)$, where x_i denotes a record within the trajectory, and n denotes the total number of records in this trajectory. Each record $x_i = (x_i.lat, x_i.lng, x_i.t, x_i.state, x_i.id)$, where $x_i.lat$, $x_i.lng$, $x_i.t$, $x_i.state$ and $x_i.id$ denote the latitude, the longitude, the timestamp, the vehicle state (e.g. busy, stop etc.), and the map-matched road ID respectively. For each trajectory, it is associated with T_a, a set of attributes which can be categorized under four classes: vehicular, spatial, temporal, and weather. Vehicular attributes provide information about the vehicle that travels the trajectory, e.g. vehicle ID. The weather attributes include information like rainfall level in the city. The temporal attributes include periodic information like time of day (at which T_x starts) and day of week. Finally, the spatial attributes are geospatial information like the total distance (traveled by T_x). Note that a trajectory is obtainable from different vehicle types, e.g. taxi and bus, and thus may exhibit different spatiotemporal characteristics depending on its vehicular source. As such, the vehicle ID of a trajectory acts as a unique identifier that also reveals the vehicle type. More formally, we define a trajectory data instance, which includes the trajectory and all its associated attributes, as $T = (T_x, T_a)$. In this work, we use a hybrid collection, i.e. a mixture, of trajectories obtained from more than one vehicle type.

Our objective: Given the input of a hybrid trajectory dataset built from a small trajectory dataset of one type of vehicles and a massive trajectory dataset of another type, we want to train a neural model that can estimate the travel time required by the small-dataset vehicles to complete any path within the road network, at any time of the day and on any day of the week.

3 ATTRIBUTE-RELATED HYBRID TRAJECTORIES NETWORK

In this section, we present our proposed method. First, we highlight the intuition of how our method is able to leverage hybrid trajectories to improve travel time estimation. Second, we provide an overview of our architecture along with the motivations behind the constituting modules. Finally, we describe in details the functions of each of these modules, the rationale behind their designs and how these designs contribute to effective travel time estimation.

3.1 Intuition of AtHy-TNet

A small trajectory dataset usually implies that the coverage of its vehicles is less extensive than vehicles of a massive dataset. However, if both datasets are sampled from the same geographical space, the much larger coverage of the latter may overlap with that of the former. AtHy-TNet leverages these overlaps within the hybrid dataset and the fact that vehicles within a dataset belong to the same type. The intuition is to learn the characteristics of trajectories

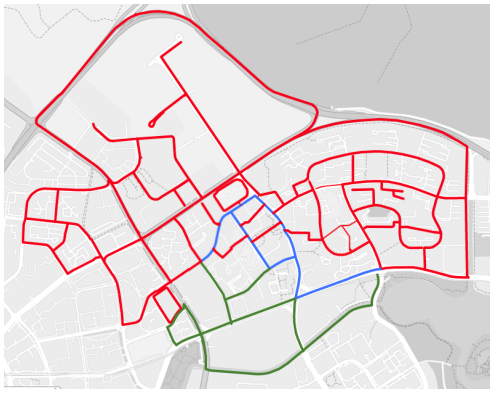

Figure 1: A small map with trajectory overlays from two vehicle types (best viewed in colour).

of both the small-dataset and big-dataset vehicles considered in the hybrid dataset, and model the correlations at parts where they overlap. The correlation learnt is then used to infer the travel time of small-dataset trajectories untraveled before, yet whose paths overlap with the big-dataset ones used during training.

For instance, Figure 1 shows a roadmap of a small district that is overlaid with trajectories from two datasets, each obtained from a different type of vehicles. Traces in green represent trajectories generated from a small dataset A, while traces in red represent trajectories obtained from a larger dataset B. Traces in blue highlight the roads where the trajectories of the two datasets overlap. As observed, trajectories from A also have a less-extensive coverage of the road network than its larger counterpart. As such, if an operator who possesses only A wishes to estimate travel time for paths in regions outside its coverage, e.g. red regions, there may be large errors since A does not consider the complexities there yet. In this case, AtHy-TNet leverages the correlations between trajectories of the two vehicle types at the blue regions and the larger coverage of B at the red regions. Specifically, it first models the multi-natured characteristics of the trajectories in both the green and red regions, while learning the correlations of these characteristics at the blue regions. This correlation can then be used to estimate the travel time at the out-of-coverage red regions.

3.2 Overview of AtHy-TNet

The architecture of the proposed AtHy-TNet is shown in Figure 2. It comprises of four modules: Attribute Correlation, Sub-path Representation, Sub-path Correlation, and Joint Prediction. The input to our model is a trajectory and its associated attributes. Path attributes are very important when dealing with hybrid data due to the different trajectory natures. However, existing work generally does not consider the correlations across these attributes nor their relative importance. As such, we design an Attribute Correlation Module to model such relationships among trajectory attributes, e.g. vehicle ID that differentiates a type of vehicles from another. We support our main path travel time estimation task with an auxiliary task of sub-path travel time estimation. The rationale is that by accurately estimating the sub-path times in a path, the path travel time can in turn be more accurately estimated. Furthermore, by modeling sub-path characteristics, it enables the model to learn

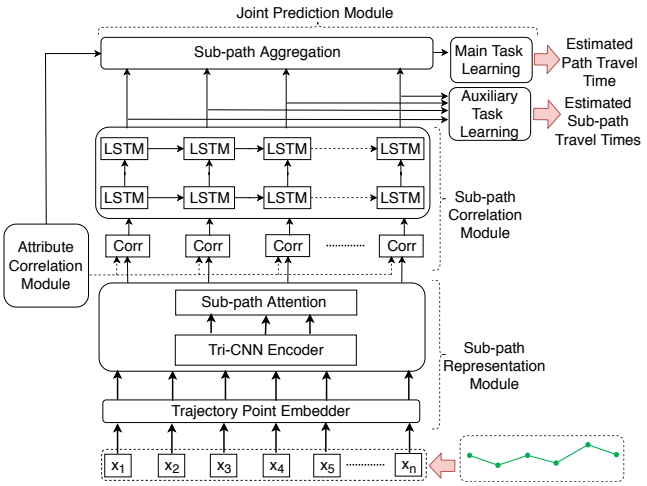

Figure 2: Architecture of AyHy-TNet Model

the trajectory differences across different vehicle types to a finer detail. To this end, we first extract the abstract spatial features of sub-paths by designing a Sub-path Representation Module. For the first time, we consider a design that effectively utilizes on-road information, e.g. map-matched road GPS coordinates. To model the temporal relationships across the sub-paths, and the correlation between each sub-path and the path attributes, we design a Sub-path Correlation Module. Finally, we leverage the Joint Prediction Module to combine the main and auxiliary task described earlier.

3.3 Attribute Correlation Module

Path travel time is largely affected by various external factors. We are the first to utilize the relative importance of each of these attributes through our proposed Attribute Correlation Module, as depicted in Figure 3. Specifically, the input here is T_a, a set of vehicular, weather, temporal and spatial attributes, i.e. vehicle ID, rainfall, day of week, time of day, and total distance. The Attribute Embedder embeds all of these attributes into higher-dimension vectors via a linear layer. For all attributes other than distance, we denote them as a_c, a_r, a_d and a_t respectively.

Global Distance Representation. Path distance is an important attribute in determining path travel time. Existing deep-learning based work focuses mainly on coarse-grained great-circle distances [26]. However, fine-grained distance data is able to reveal additional information about the path traveled. Hence, we integrate not only coarse-grained, but also fine-grained distance in our model. Specifically, we consider two attributes: a coarse-grained global trip distance, and a fine-grained global trip distance derived from on-road distance information. We convert each distance into a $\mathbb{R}^{16}$ vector, denoted as d_c and d_f, via the Attribute Embedder. Then we concatenate them and apply a linear transformation:

$$a_g = W_d[d_c, d_f] + b_d, \tag{1}$$

where a_g is the global distance representation and W_d and b_d are the parameters of this operation. The square brackets denote the concatenate operator.

Attribute Attention. Certain attributes are relatively more important than others on some occasions in revealing the travel time.

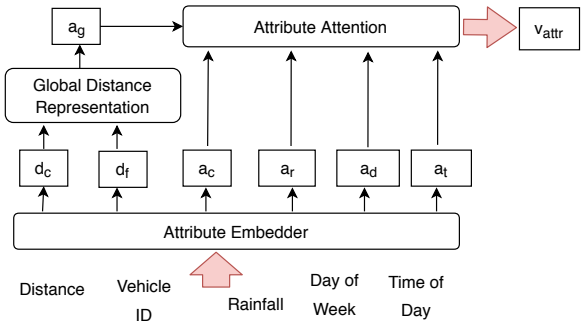

Figure 3: Attribute Correlation Module. a_c, a_d, a_r and a_t represent embedded vectors for vehicle ID, day-of-week, rainfall, and time-of-day respectively. d_c (resp. d_f) represent the fine (resp. coarse) grained embedded distance vectors.

Vehicle ID is generally more important when dealing with hybrid data because the trajectory of a type of vehicles may differ greatly from the trajectory of another, e.g. public buses move much slower than taxis. A vehicle traveling in the downtown during peak hours will be slow, due to traffic congestion. In this case, the day and time of the day are more important attributes. Since attention mechanism is a powerful way to capture such correlations, we leverage it here, as follows:

$$z_i = a_i c_i \tag{2}$$

$$\alpha_i = \frac{\exp(z_i)}{\sum_{i \in A} \exp(z_i)} \tag{3}$$

$$v_{\text{attr}} = \sum_{i \in A} \alpha_i a_i, \tag{4}$$

where α_i is the weight for the attribute vector a_i, $A = \{c, d, t, r, g\}$. Parameter c_i is used to calculate an intermediate score z_i. We characterize our attention mechanism with $\{a_i\}$ and $\{c_i\}$, and v_{attr} denotes the output of our attention mechanism and this module.

3.4 Sub-path Representation Module

The Sub-path Representation Module is the encoder for the input trajectory. First, the Trajectory Point Embedder is used to embed the trajectory point information to a higher-dimension representation. Then, the Tri-CNN Encoder, which is an encoder with three-channel output, is utilized to extract high-level abstract sub-path features based on roads used in the trajectory. To attend to the important channel of sub-path features, we design a Sub-path Attention mechanism, inspired by [1, 29]. Finally, we use a Local Distance Representation encoder to create a more accurate representation for the sub-path distance.

Trajectory Point Embedder. The input of this module is a sequence of trajectory points T_x. For each point x_i in T_x, we map-match it to a road using [23] and embed its road ID to a vector. Due to the fact that there are often thousands of roads in large cities, using traditional one-hot encoding results in a vector too large. As such, we use the idea of condensed word vector from natural language processing and convert it to $\mathbb{R}^{32}$ for an accurate representation while saving space. Since the GPS coordinates of each point reveal fine-grained information of the vehicle, we seek to ensure that their impact are not buried by the large embedded road ID vector. This is achieved by converting a point's latitude to $\mathbb{R}^{16}$, and doing the same for its longitude. We embed the vehicle status to $\mathbb{R}^2$. We then concatenate the embedded vectors and convert it to $\mathbb{R}^{16}$ using a non-linear layer, as follows:

$$u_a = \tanh(W_c x_i + b_c), \tag{5}$$

where W_c and b_c are the parameters of the embedder.

Tri-CNN Encoder. This encoder is used to extract high-level abstract sub-path features, while mitigating the noise incurred during the recording process. The rationale is that trajectory recording is a noisy process that often results in imprecise GPS coordinates recorded. Map-matching, on the other hand, is capable of estimating precisely the roads used during the trajectory. As such, by integrating the GPS coordinates of its map-matched road, it helps in extracting high-level abstract sub-path features in a more accurate fashion. To this end, we create u_s and u_e using the same procedure as Equation 5. The difference is that instead of using the $x_i.lat$ and $x_i.lng$ as inputs during the creation of u_s (resp. u_e), the latitude and longitude of the start (resp. end) point of the road x_i belongs to are used.

Inspired by DeepTTE [26], we then utilize a 1D convolutional neural network (CNN) to create a spatial representation for the sub-paths in u_a. CNNs have been widely used in various image recognition tasks [20, 22] for their power in capturing spatial and temporal dependencies. For tasks where the input is a 1D-sequence, e.g. sentence classification [14], the 1D-CNN is very effective in extracting local subsequences of data instances and derive patterns within each subsequence. As such, it is applicable to our context of path travel time estimation, where the 1D input sequence (resp. subsequence) is a trajectory (resp. sub-path) and each GPS point is a data instance. In particular, 1D-CNN is effective in capturing finer-grained information like turns at road junctions [26]. Typically, a 1D-CNN contains a filter of a fixed width that slides across the data instances to capture patterns within each sub-sequence. In our model, the formulation is as follows:

$$v_a = \text{ELU}(W_a * u_a), \tag{6}$$

where $W_a \in \mathbb{R}^{k \times d \times 16}$ is the parameter, $*$ is the convolution operation, k is the filter width and d is the output dimension. ELU is the activation function [4], where $\text{ELU}(x) = \exp(x) - 1$ for $x \leq 0$ and x for $x > 0$. We then use the same 1D-CNN to create another sub-path spatial representation v_s (resp. v_e) from u_s (resp. u_e) respectively.

Sub-path Attention. Different CNN outputs may have different importance. If T is very noisy, a greater importance should be assigned to v_s or v_e, instead of v_a. As such, we pass the output of the Tri-CNN Encoder through our Sub-path Attention mechanism. This attention mechanism is similar in spirit to the Attribute Attention proposed in Section 3.3, with one exception. The formulation is as follows:

$$y_i = \sum_{j=1}^{n-k} v_{i,j} f_i \tag{7}$$

$$\beta_i = \frac{\exp(y_i)}{\sum_{i \in S} \exp(y_i)} \tag{8}$$

$$v_m = \sum_{i \in S} \beta_i v_i, \tag{9}$$

where β_i is the weight for v_i, $S = \{s, a, e\}$, and $\sum_{i \in S} \beta_i = 1$. To learn β_i, we use a parameter vector f_i. Note that unlike Attribute Attention, the intermediate attention score y_i is calculated by the sum of the inner product between $v_{i,j}$ and f_i across all j, where $v_{i,j}$ denotes the j-th sub-path vector of v_i.

Local Distance Representation. The sub-path distance is an important information. As mentioned in Section 3.3, supporting it by the integration of fine-grained distance may create a more accurate representation of the distance traveled. To this end, we consider two inputs, the coarse-grained local distance of the sub-paths and its fine-grained version. We denote these by $l_c, l_f \in \mathbb{R}^{n-k}$ respectively. Similar to Section 3.3, we convert each of l_c and l_f into $\mathbb{R}^{(n-k)\times 16}$ using a linear layer. Then, we perform a concatenation on l_c and l_f, and convert it to $\mathbb{R}^{(n-k)\times 16}$ using another linear layer. Finally, we concatenate this vector with v_m of the Spatial Representation Attention. We denote the final output of this module as m.

3.5 Sub-path Correlation Module

Correlation Component. Travel time is highly dependent on the path attributes, as mentioned earlier. To capture the inter-dependencies between the attributes and the sub-paths, we design a Correlation component where v_{attr} is first concatenated with the representation of each sub-path in m, before passing through a linear layer, as follows:

$$s_i = W_s[m_i, v_{\text{attr}}] + b_s, \tag{10}$$

where m_i is the representation of the i-th sub-path, $1 \le i \le n-k$, and W_s and b_s are the parameters of this operation.

LSTM Layer. To further capture the temporal dependencies among sub-paths, we feed the results into long short-term memory (LSTM) network, a type of recurrent neural network (RNN). Generally, RNNs enable the processing at every step of a sequential input to leverage information from the previous steps. In particular, LSTM is capable of learning longer-term dependencies by using a memory cell, and resolving the issue of vanishing and exploding gradient in standard RNNs. Briefly speaking, the hidden state h_t and memory cell c_t are function of h_{t-1} and c_{t-1} from the previous step, and input vector s_t, or formally:

$$c_t, h_t = \text{LSTM}(c_{t-1}, h_{t-1}, s_t), \tag{11}$$

LSTM uses an input and a forget gate to control the information flow within the recursive operation. We also use two stacked LSTM to allow greater model complexity. For more details about LSTM, we refer to [10]. We denote the output of this layer as h_{sub}.

3.6 Joint Prediction Module

This module jointly learns the main task of path travel time estimation, with an auxiliary task of sub-path travel time estimation, inspired by [2, 26]. The intuition is that the travel time of a path is essentially made up of the sum of its sub-path travel times. By accurately estimating the sub-path times, the path travel time can be more accurately predicted, vice versa.

Sub-path Aggregation. Due to length variability of h_{sub}, we convert it to a fixed-length vector via pooling. While mean-pooling treats all sub-paths equally, attention-pooling gives more attention to critical sub-paths, e.g. many traffic lights or road segments [26]. As such, we utilize an attention mechanism that is similar in structure as Attribute Attention of Section 3.3. However, instead of an arbitrary parameter vector, we convert v_{attr} via a linear layer to a vector that is similar in size as a vector in h_{sub}, and use it as the parameter. The output of the aggregation is denoted by r_{path}.

Main and Auxiliary Task Learning. To generate the estimated path travel time, we use deep residual learning. Deep residual learning is a technique for training of very deep neural networks and has shown superior performance across multiple recognition [9] and certain geo-spatial tasks [34]. The intuition is that each residual unit focuses on fine-tuning the output from the previous unit by only learning an added "residual". This is achieved through shortcut connections across the units. Such designs mitigate the degradation issue of deep neural networks where adding more layers results in lower accuracies, and makes it very effective to train deep networks for maximal representation power. Specifically, we pass the concatenated vector $[r_{\text{path}}, v_{\text{attr}}]$ through a linear layer, followed by a series of residual units. The formulation of a residual unit is:

$$r_l = r_{l-1} + \text{ReLU}(W_r r_{l-1}), \tag{12}$$

where r_{l-1} denotes the output from the $(l-1)$-th residual unit. Also, $1 \le l \le L$, where L is the total number of residual units. ReLU is the activation [6]. W_l is the learnable parameters of this unit. Finally, we pass r_L through a linear layer to get the estimated path travel time $t_{\text{path}} \in \mathbb{R}^1$. The training objective for this task is:

$$C_{\text{path}} = \left| \frac{t_{\text{path}} - g_{\text{path}}}{g_{\text{path}}} \right|, \tag{13}$$

where g_{path} is the ground-truth path travel time.

To get the estimated travel time $t \in \mathbb{R}^{(n-k)\times 1}$ for the sub-paths, we simply pass h_{sub} through several fully-connected linear layers. The objective for this task is:

$$C_{\text{sub}} = \frac{1}{n-k} \sum_{i=1}^{n-k} \left| \frac{t_i - g_i}{g_i} \right|, \tag{14}$$

where t_i (resp. g_i) denotes the estimated (resp. ground-truth) value for the i-th sub-path.

For optimization during training, we use the following loss function:

$$C_{\text{join}} = \zeta C_{\text{path}} + (1 - \zeta) C_{\text{sub}}, \tag{15}$$

where ζ is a weighing factor that considers the relative importance of the error of each task.

4 EXPERIMENTS

To ascertain the effectiveness of our proposed method, we conduct experiments on real-world datasets. Specifically, we focus on the application of estimating public bus travel time using a hybrid dataset of public bus and taxi trajectories.

4.1 Experimental Set-up

This section describes the datasets used, the baselines we consider for comparisons, and the evaluation metrics.

Datasets. For a fair validation process and to construct a hybrid trajectory dataset, we require datasets obtained from the same geographical space. As such, we make use of the following datasets:

- **Singapore Smart Card Dataset:** This dataset contains passenger trip records obtained from CEPAS card records from Singapore's public bus network. After pre-processing, there are 204,000 trajectories in all. We explain the preprocessing steps below.
- **Singapore Taxi Dataset:** We use a dataset that has 11.2M taxi trajectories. Each point within the trajectory contains the GPS coordinates, the time instance at which it was sampled, and the status of the taxi (i.e. busy, idle, stop, on-call, or hired) at that time instance. Each trajectory also includes the following information: a vehicle ID, time of day, and day of week. The longest trajectory has 128 points.

We further collect city-scale rainfall records; a total of 5 distinct levels, from Singapore. All the above datasets are obtained over an overlapping time period in March 2016.

Pre-procesing: Bus Trajectory Derivation from Smart Card Dataset. A passenger trip record in the smart card dataset consists of the following features: origin stop, destination stop, bus number, tap-in time, and tap-out time. Specifically, tap-in (resp. tap-out) time refers to the time at which the passenger boards (resp. alights) the bus at origin (resp. destination) stop by tapping in (resp. out) at the on-bus fare gantry. A bus journey starts from a bus terminal, passes a series of scheduled bus stops, and ends at a bus terminal. Unlike taxis, public buses generally follow scheduled routes. Through basic sorting and processing, we derive a trajectory for each bus journey in the dataset, where each point within the trajectory contains the GPS coordinates of the bus stop, as well as the timestamp of the first tap-in at that stop during the journey. We obtain the exact on-road distance between any two consecutive stops traveled by the bus from the publicly accessible bus schedules. As such, we can further derive the on-road distance between any two consecutive GPS points of the trajectory, as well as the total on-road journey distance. Since the date and time of the journey are recorded, we can get the following attributes required by our model: day of week, and time of day. Finally, instead of using the vehicle states of taxis, we assign a constant bus state to each GPS point of the bus trajectory to further differentiate bus trajectories from taxi ones.

Data Characteristics. Table 1 shows some statistics of the trajectory datasets. Observe that the typical bus data is vastly different from the taxi data. The mean, 25%-ile and 75%-ile travel times show that the total trip duration of a bus trajectory is typically several times that of a taxi's. Bus travel time also has a larger variance, and the time-gaps between consecutive trajectory points are larger. Moreover, the points in a taxi trajectory are sampled at a regular frequency, but the points in a bus trajectory are sampled when the bus is at the bus stops.

Training, Validation and Testing Sets. We create a bus training set Bus-Training by sampling 70% of the bus lines in Singapore Bus Dataset and collecting all the trajectories of these lines. Similarly, we sample 10% (resp. 20%) of the remaining bus lines for the validation (resp. testing) data, denoted as Bus-Validation (resp. Bus-Testing). We also create a taxi training set Taxi-Training, by randomly sampling from the Singapore Taxi Dataset a number of trajectories that is 5 times that of the number of trajectories in Bus-Training. Finally, we create a hybrid dataset Hybrid-Training by combining Bus-Training and Taxi-Training. This is done by simply putting the bus and taxi trajectories in a common dataset. For example, if Bus-Training = $\{T_1, T_2\}$ and Taxi-Training = $\{T_3, T_4, T_5\}$, where each $T_i, 1 \leq i \leq 5$ denotes a trajectory instance, Hybrid-Training = $\{T_1, T_2, T_3, T_4, T_5\}$. We repeat the above steps for 5 times to get 5 different sets of data.

Table 1: Trajectory Data Characteristics

Statistical Index	Bus	Taxi
Travel Time Mean (s)	3,245	692
25%ile Travel Time (s)	1,841	283
75%ile Travel Time (s)	4,192	933
Travel Time Std (s)	1,756	573
Time Gap Mean (s)	204	34
Sampling Interval (s)	-	30

Evaluation Metrics. For our experiments, we adopt the following evaluation metrics: mean absolute percentage error (MAPE), mean average error (MAE), and root mean square error (RMSE).

Implementation Details and Parameter Setting. We implement our model on Pytorch 0.4.1. Our model training/evaluation is done on Nvidia Tesla M40 GPU, and Linux machine with an Intel Xeon 2.6GHz CPU and 64GB RAM. We use Adam optimization algorithm to fine-tune the parameters. The learning rate we use is 0.001 and we train our model in batches of 400 over 100 epochs. For hyper-parameters settings, we set $k = 3$ and $c = 32$ in the Sub-path Representation Module. The embedding size of the attributes or inputs are chosen based on empirical testings. In the LSTM layer of our Sub-path Correlation Module, we use the hidden vector size of 128. In the Correlation component, we use an output size that is the same as the input. In the Joint Learning Module, we set the number of residual units used as 3; the size for each unit is 128. We also set the number of fully-connected layers used in the auxiliary task prediction as 1. Finally, we set ζ as 0.7, for a stronger focus on the main task loss function. To provide sufficient clout for the relatively smaller bus trajectory sets, we also associate a constant vehicle ID for all the bus trajectories.

Baselines. We consider several baselines in our experiment. Each baseline has a superscript $*$, † or ‡, which denote that the model uses Bus-Training, Taxi-Training, Hybrid-Training for training respectively. In all cases, Bus-Validation and Bus-Testing are used for validation and testing. The baselines are as follows:

- **DeepTTE*** As there is no prior work on hybrid-trajectory-based estimation, we compare with the state of the art single-source method. DeepTTE [26] is chosen as it is the best-performing one to date, yet is also modifiable to support hybrid trajectories. Specifically, instead of considering the driver ID feature in this baseline, we use vehicle ID to differentiate bus trajectories from taxi ones. Instead of using the weather feature that is required in DeepTTE, we replace that feature with the rainfall level present in our datasets. In this baseline, only Bus-Training is used for model training to demonstrate the effectiveness when only a single-source small trajectory dataset is available.

Table 2: MAPE, MAE, and RMSE on AtHy-TNet and the baselines. Best results are in bold face and second best underlined.

Method	MAPE (%)	MAE (s)	RMSE
DeepTTE*	15.16	367.16	374.51
DeepTTE†	41.93	1244.69	1433.67
DeepTTE‡	13.15	350.60	358.26
LightGBM‡	14.60	393.83	544.25
HyMLP‡	17.40	497.70	608.60
HyLSTM‡	14.81	388.70	406.70
AtHy-TNet*	13.66	329.45	342.0
AtHy-TNet-a‡	<u>10.91</u>	<u>288.32</u>	<u>295.73</u>
AtHy-TNet‡	**10.24**	**264.21**	**273.12**

- **DeepTTE**† This baseline is similar to DeepTTE* except that Taxi-Training is used for training. It is chosen to explore the effectiveness of solely utilizing a larger trajectory dataset, but of different characteristics.
- **DeepTTE**‡ This baseline is similar to the previous two, except that Hybrid-Training is used for training. This baseline serves to show how estimation changes when the training dataset uses not only a small single-source dataset, but also a larger one obtained from the same geographical space.
- **LightGBM**‡ LightGBM is one of the most efficient and high-performing gradient boost decision tree method [13]. The trajectory input for this baseline is the same as AtHy-TNet. Apart from the total path distance, the attributes of the trajectory are specified as categorical features in the program.
- **HyMLP**‡ This baseline is a simple 5-layer fully-connected network with ReLU activation. The input is the same as AtHy-TNet and each categorical attribute in the input is embedded to a low-dimensional vector.
- **HyLSTM**‡ This baseline uses a 2-stacked LSTM. The input is the same as AtHy-TNet, except that neither sub-path representation nor attribute correlation is considered. Similar to HyMLP‡, each categorical attribute is embedded to a low-dimensional vector, before feeding into the LSTM. The estimation is obtained by mean-pooling the LSTM output and passing through residual fully-connected layers.
- **AtHy-TNet*** This baseline uses AtHy-TNet on Bus-Training instead of Hybrid-Training. The purpose is to demonstrate the raw improvements brought about by our model when a single-source small dataset is used.
- **AtHy-TNet-a**‡ This baseline is a simplified model of AtHy-TNet. Instead of using the Sub-path Representation Module, AtHy-TNet-a‡ uses the Geo-Conv Layer proposed in [26].

4.2 Evaluation Results

We conduct experiments on the five sets of {Bus-Training, Taxi-Training, Hybrid-Training, Bus-Validation, Bus-Testing} described earlier, and record the average testing results on Bus-Testing in Table 2. As observed, methods that utilize hybrid trajectories perform better than their single-source versions. For example, the performance of DeepTTE‡ is at least 2% stronger than DeepTTE* and DeepTTE †. Despite the fact that Taxi-Training is much larger than Bus-Training and supposedly not as affected by data sparsity, performance of DeepTTE † is significantly worse than DeepTTE*. This shows that simply replacing a dataset with another results in large errors, due to the vast differences across different sources. AtHy-TNet-a‡ shows a stronger performance than all other hybrid-data methods by at least 2%. This demonstrates effectiveness of our model in estimating across hybrid trajectories, and by considering correlations of the relevant attributes. The performance of AtHy-TNet‡ is a further improvement from AtHy-TNet-a‡ by around 0.7%. This demonstrates the benefits of extracting high level sub-paths spatial features through fine-grained road-based information. As such, our experiment results show that the effectiveness of our model in learning multi-nature correlations of hybrid trajectory data. Note that AtHy-TNet* demonstrates a performance stronger than DeepTTE* by 1.5%, which shows that our method is effective even when single-source training data is used.

Comparison with Google Maps. We also compare AtHy-TNet with Google Maps API [7]. Adopting the treatment in [27], we query Google Maps Directions API at the same time and weekday as the starting time of each trip, by assuming a strong weekly periodicity. Since the API does not support the querying of past trips, all the queries correspond to the current or future time. As arbitrary public bus routes outside of the existing bus lines are unsupported in the query, the travel mode is set as Driving. Due to credit limits, 30k trips within a set of Bus-Testing are randomly chosen for comparison. The MAPE, MAE, and RMSE are 20.9%, 491, and 647 respectively. In contrast, the errors are 11.24%, 237, and 241.3 respectively when the same trips are estimated using AtHy-TNet. A reason why Google Maps does not perform as well may be that the data sources for its estimations in the Driving mode are largely based on non-public-bus vehicles. Since the travel time of arbitrary public bus routes are unsupported in online map services, it further highlights the necessity of a solution for vehicles with small trajectory datasets.

Effect of Varying Taxi and Bus Data Size in Hybrid-Training. Next, we keep the bus trajectories in Hybrid-Training constant and vary the concentration of taxi trajectories. Specifically, we consider taxi data that is K times the size of the bus data. We evaluate the performance of AtHy-TNet‡ over different values of K, using Bus-Validation and Bus-Testing. The results are shown in Figure 4. As observed, the performance is significantly worse when no taxi trajectories are used in the hybrid training data, but improves rapidly when they are introduced. As early as $K = 1$, the decrease in error is more than 2%. The performance strengthens with increasing K and settles at its peak from $K = 4$ onward. The maximum achievable improvement is nearly 4%. This demonstrates benefits of hybrid data, and capacity of AtHy-TNet‡ to effectively utilize hybrid data, even when the external data used is small. Error tends to be larger when K is small, since a smaller external taxi data often implies a smaller extended area coverage. As such, estimation does not improve as much as when larger taxi data/coverage is considered in the hybrid data.

We also vary the concentration of bus trajectories while keeping the taxi trajectories in Hybrid-Training constant. The results are

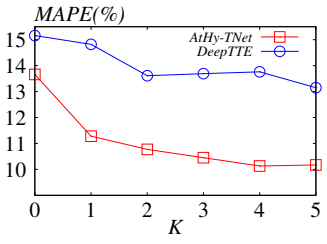

Figure 4: Performance over different K values

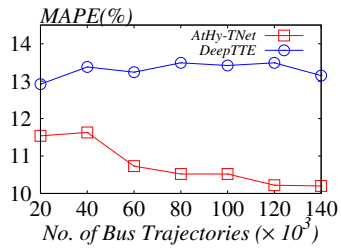

Figure 5: Performance over different bus data size

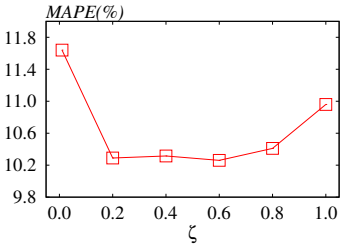

Figure 6: Performance of AtHy-TNet over different ζ

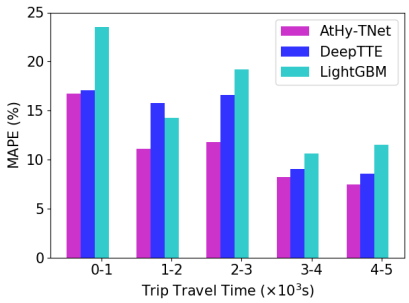

Figure 7: Performance over different travel times

shown in Figure 5. As observed, the accuracy improves as the bus trajectory data size increases. The reason is that when the bus data is small, the area coverage of buses that can be used for correlation-learning with taxis tends to be small too. As such, the error is larger as compared to when more bus data is available for correlation-learning. Nonetheless, AtHy-TNet‡ demonstrates a satisfactory accuracy using bus data sizes even as small as 20k and 40k. This shows that our method is capable of leveraging external data for estimation to address the inaccuracy brought about by a single-source small dataset. Moreover, in both cases, AtHy-TNet‡ shows a stronger performance and a steeper descent over increasing data size than DeepTTE‡. This shows the suitability of our model over prior art in effectively modeling across hybrid trajectories.

Performance across Different ζ in the Joint Learning Module. Figure 6 shows how the performance of AtHy-TNet varies across different values of ζ, from 0.01 to 1.0. As observed, the performance is satisfactory in general across all values. Although the performance remains around a constant level in the middle of the range, the errors are slightly larger at the ends, i.e. 0 and 1.0. This demonstrates the benefits of learning the main task of path travel time estimation with an auxiliary one and considering the errors of both during training.

Performance across Different Trip Travel Times. Figure 7 compares the model performance across trajectories of different trip durations. Generally, AtHy-TNet consistently shows a stronger performance than DeepTTE and LightGBM across all trip durations. It is interesting to note that the error tends to decrease as trip duration increases for all methods. The reason may be that unlike a short trip, a long trip usually has more constituting sub-paths. Since each sub-path provides valuable information that is useful for estimation, having more sub-paths means the longer trips have more information usable by the methods. This results in the travel time of these trips to be estimated more effectively than the shorter ones. Furthermore, the rate of error-decrease for AtHy-TNet is the fastest as compared to other methods. Specifically, error decreases from 16% to around 11% when the trip duration increases from 0-1000s to 1000-2000s, stays constant at 2000-3000s, before further decreasing to around 8% for durations within 3000-5000s. In contrast, the errors of DeepTTE stay nearly constant during the shorter trips and only starts to decrease towards the longer ones (3000-5000s).

Performance across Different Days of the Week. Figure 8 shows the performance of AtHy-TNet and baselines across the different days of the week where 1 represents Monday and 7 represents Sunday. As observed, AtHy-TNet out-performs DeepTTE and LightGBM across all days. It is interesting to know that the trend of errors' variation across the week is similar across all three methods. Specifically, error increases almost linearly from Monday to Sunday, with dips on Wednesday and Friday. Estimation errors are higher on weekends than weekdays since the travel patterns of passengers are less regular. It may also be because the number of trajectories collected on weekends is generally smaller as there are fewer weekend days in a week. The rate of increase for AtHy-TNet, as shown by the gradient of the slope, is slow as compared to the other two methods. Furthermore, its errors fall within the satisfactory performance range of 9-11%. This demonstrates the performance robustness of AtHy-TNet across different days of the week.

Performance across Different Times of the Day. We also evaluate the estimation performance across different times of the day. We conduct the test separately for weekdays and weekends, since the travel characteristics of weekdays and weekends are largely different. The results are shown in Figure 9 and 10 respectively, based on the 24-hour clock. As observed, AtHy-TNet out-performs the baselines across all times of the day for both cases. The errors of AtHy-TNet also show a smaller variance in both cases. Similar to Figure 8, the MAPEs during the weekends are mostly higher than during the weekdays. For both weekdays and weekends, errors remain relatively constant for all methods during the later parts of the day, i.e. 9 hrs onwards. Note that the error of the earliest period, i.e. 0-3 hrs, of weekdays is higher than that of the later periods. In contrast, the error over the same time period during weekends is lower than the later ones. The reason for such discrepancy may be because there are fewer bus services that run at that period. As such, there are much fewer trajectories available for training and testing, and this results in a large variance in errors between weekdays and weekends. Furthermore, specific to AtHy-TNet, the error during the 6-9 hrs of weekdays is the highest as compared to the rest of the day. This is reasonable since the 6-9 hrs period is the morning peak hours when traffic congestions are rampant, due to people going

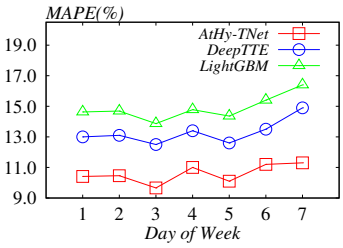

Figure 8: Performance over different days of the week

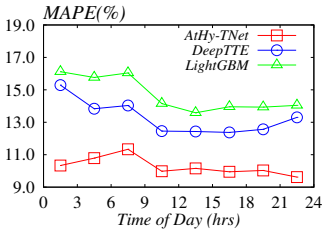

Figure 9: Performance over different times of the day (weekday)

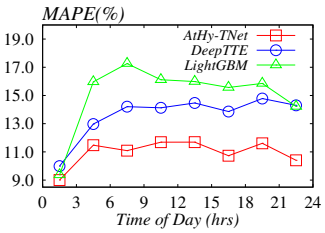

Figure 10: Performance over different times of the day (weekend)

to a common set of places. As a result, the difficulty of estimation is higher. Compared to the baselines, this observation also shows that our model manages to reveal such temporal-dependencies in the datasets.

Attribute/Feature Importance Evaluation. We study the effect of each of the attributes used in AtHy-TNet, by evaluating the performance of AtHy-TNet when only that attribute is excluded from the model. When the day-of-week attribute is excluded, the MAPE worsens by 0.46%. When time-of-day attribute is excluded, the MAPE worsens by 0.99%. This is aligned to our intuition that traffic complexities, e.g. traffic congestions, are different in different times of the day/week. As such, temporal attributes are important to model such differences. When the rainfall attribute is excluded, MAPE increases by 0.43%. This is coherent with the well-studied fact that vehicles tend to move slower on rainy days. To study the effect of differentiating bus trajectories from taxi ones, we exclude vehicle ID. Recall that the bus states used are generally different from the taxi ones. As such, we also exclude the vehicle state of each GPS point. As a result, the MAPE worsens by 0.96%. This highlights the importance of differentiation of vehicle types when dealing with hybrid trajectories, since vehicles of different types may have distinct travel behaviours.

Predicting Time. The predicting time of AtHy-TNet is fast, in spite of its slower training time. The time taken to estimate a batch of 400 trajectories is around 0.023s.

4.3 Transfer Applications and Limitations.

AtHy-TNet is the first work that demonstrates the benefit of using hybrid dataset for a task. A possible transfer application of the method is to use it for the trajectory classification problem of a vehicle type that has a small dataset, e.g. classify bus trajectories based on the degree of in-bus crowdedness. Due to privacy and legal concerns, it is often hard for one operator, e.g. taxi company, to share its data with another, e.g. bus company. As such, another potential extension is the use of federated learning for the training process. For example, instead of training on a hybrid dataset, the model may first train on the taxi dataset, before fine-tuning by training another time on the bus dataset. In this case, only the model needs to be shared among the companies while the datasets are kept private.

A limitation of AtHy-TNet is that it is a preliminary attempt at demonstrating the effectiveness of using hybrid-trajectory data. As such, the model can be improved in several ways, like include finer-grained road information such as regulated speed limits, and if the road has electronic road tolls installed. Furthermore, AtHy-TNet deals with the estimation of a single type of vehicles when the single-source dataset is small. An extended exploration on leveraging hybrid datasets for multi-modal path travel time estimation may be useful, to public organizations for traffic studies, and to commuters for multi-modal trip planning.

5 RELATED WORK

Travel time estimation using trajectory data are generally classified into two categories: 1) Link-based, and 2) Sub-path-based methods.

For the link-based methods, travel time estimation is carried out for the individual road segments [12, 18, 19, 25, 28, 31], before these times are combined to form the travel time of the whole path. Yang et al. utilize spatio-temporal Markov models to model traffic behaviours and forecast the travel time at all the road segments [31]. In [25], Wang et al. make use of regression tree and probabilistic graphical methods based on some correlations observed from the traffic data. In [28], the authors propose a deep learning framework that predicts traffic speed of road segments using spatio-temporal information of contiguous road segments. Rahmania et al. propose a fixed point formulation that simultaneously infers route choice and travel time [19]. Jenelius et al. [12] use a multivariate probabilistic principal component analysis model that is able to model correlation patterns between days. One limitation about such methods is that they do not consider certain travel complexities on the path level, such as traffic lights, and turn costs.

Sub-path-based methods mitigate this limitation to some extent [5, 16, 30, 32]. The key idea is that path travel times can be estimated by considering historical path/sub-path travel times, which implicitly consider costs that are not modeled in the link-based methods. In [32], Yuan et al. propose a time-dependent landmark graph to model the time-varying traffic patterns, and techniques to compute the fastest path based on this graph and the experience of drivers. In [30], the authors utilize a tensor decomposition approach to resolve the issue of data sparsity, and a dynamic programming method to find the optimal concatenation of trajectories for path travel time estimation. Dai et al. propose the hybrid graph paradigm and techniques to address the issue of non-standard travel time distribution, data sparseness and the dependencies among roads travelled during a path [5]. Li et al. further consider scenarios where only a small number of probe vehicles is used to collect the data and travel times of the same path may have large variances [16]. More recently, deep learning has shown further success for this task. Wang et al. propose DeepTTE, which models the path travel time of taxis in an end-to-end fashion, while considering

the spatio-temporal and attribute characteristics [26]. Zhang et al. partition the road network into grids, and consider various short and long-term features [33].

On the broader scale, there also exists work that estimates travel time given the input of an origin-destination (OD) pair [11, 15]. However, while these methods are advantageous in faster computations, they have a limited consideration in terms of the vehicle path choice variabilities between the origin and destination, which the travel time of a vehicle is highly dependent on. Furthermore, deep learning has been used in various spatio-temporal problems, such as crowd-flow prediction [34], transportation mode prediction [21] and trajectory predictions for road agents [3, 8, 17, 24, 35]. In general, utilizing deep learning for path travel time estimation is a new field, and there has not been any work which seeks to perform this task across hybrid trajectories.

6 CONCLUSION

In this paper, we propose an Attribute-related Hybrid Trajectories Network capable of estimating path travel times by effectively utilizing hybrid trajectory data. This method implicitly models the correlations across hybrid trajectories, something largely ignored in existing work. It also extracts the correlations of attributes that affect path travel time and leverage on-road information to extract sub-path features. Through experiments, we demonstrate the superior performance of our method over the state of the art methods. For future work, we will explore how path travel time estimation can be used in identifying real-time traffic conditions.

7 ACKNOWLEDGEMENTS

This work is supported in part by National University of Singapore under the SUG Grant. It is also supported in part by MOE, Singapore under grant MOE2015-T2-2-069. We gratefully acknowledge the support of NVIDIA AI Tech Center (NVAITC) for our research.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[3] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. 2018. TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions. *arXiv preprint arXiv:1812.04767* (2018).
[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).
[5] Jian Dai, Bin Yang, Chenjuan Guo, Christian S Jensen, and Jilin Hu. 2016. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment* 10, 3 (2016), 85–96.
[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.
[7] Google. 2019. Google Maps Platform. https://developers.google.com/maps/documentation/
[8] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2255–2264.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[11] Jilin Hu, Chenjuan Guo, Bin Yang, Christian S Jensen, and Lu Chen. 2018. Recurrent Multi-Graph Neural Networks for Travel Cost Prediction. *arXiv preprint arXiv:1811.05157* (2018).
[12] Erik Jenelius and Haris N Koutsopoulos. 2018. Urban network travel time prediction based on a probabilistic principal component analysis model of probe data. *IEEE Transactions on Intelligent Transportation Systems* 19, 2 (2018), 436–445.
[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3146–3154.
[14] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[15] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *International Conference on Knowledge Discovery and Data Mining,(KDD)*.
[16] Yang Li, Dimitrios Gunopulos, Cewu Lu, and Leonidas Guibas. 2017. Urban Travel Time Prediction using a Small Number of GPS Floating Cars. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 3.
[17] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. 2018. TrafficPredict: Trajectory prediction for heterogeneous traffic-agents. *arXiv preprint arXiv:1811.02146* (2018).
[18] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. 2012. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 595–604.
[19] Mahmood Rahmani, Haris N Koutsopoulos, and Erik Jenelius. 2017. Travel time estimation from sparse floating car data with consistent path inference: A fixed point approach. *Transportation Research Part C: Emerging Technologies* 85 (2017), 628–643.
[20] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[21] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. 2016. DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level.. In *IJCAI*, Vol. 16. 2618–2624.
[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
[23] Youze Tang, Andy Diwen Zhu, and Xiaokui Xiao. 2012. An efficient algorithm for mapping vehicle trajectories onto road networks. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 601–604.
[24] Anirudh Vemula, Katharina Muelling, and Jean Oh. 2018. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–7.
[25] Dong Wang, Wei Cao, Mengwen Xu, and Jian Li. 2016. Etcps: An effective and scalable traffic condition prediction system. In *International Conference on Database Systems for Advanced Applications*. Springer, 419–436.
[26] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. AAAI.
[27] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 19.
[28] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic speed prediction and congestion source exploration: A deep learning method. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 499–508.
[29] Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*. 606–615.
[30] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 25–34.
[31] Bin Yang, Chenjuan Guo, and Christian S Jensen. 2013. Travel cost inference from sparse, spatio temporally correlated time series using Markov models. *Proceedings of the VLDB Endowment* 6, 9 (2013), 769–780.
[32] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2013. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Trans. Knowl. Data Eng.* 25, 1 (2013), 220–232.
[33] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. DeepTravel: a Neural Network Based Travel Time Estimation Model with Auxiliary Supervision. *arXiv preprint arXiv:1802.02147* (2018).
[34] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.. In *AAAI*. 1655–1661.
[35] Jing Zhao, Jiajie Xu, Rui Zhou, Pengpeng Zhao, Chengfei Liu, and Feng Zhu. 2018. On Prediction of User Destination by Sub-Trajectory Understanding: A Deep Learning based Approach. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1413–1422.