CompactETA: A Fast Inference System for Travel Time Prediction

Kun Fu, Fanlin Meng, Jieping Ye, Zheng Wang
AI Labs, Didi Chuxing
Beijing, China
{fukunkunfu,mengfanlinmengfanlin,yejieping,wangzhengzwang}@didiglobal.com

ABSTRACT

Computing estimated time of arrival (ETA) is one of the most important services for online ride-hailing platforms like DiDi and Uber. With billions of service queries per day on such platforms, a fast inference ETA module ensures the efficiency of the overall decision system to guarantee satisfied user experience, as well as saving significant operating cost. In this paper, we develop a novel ETA learning system named as CompactETA, which provides an accurate online travel time inference within 100 microseconds. In the proposed method, we encode high order spatial and temporal dependency into sophisticated representations by applying graph attention network on a spatiotemporal weighted road network graph. We further encode the sequential information of the travel route by positional encoding to avoid the recurrent network structure. The properly learnt representations enable us to apply a very simple multi-layer perceptron model for online real-time inference. Evaluation of both offline experiments and online A/B testing verifies that CompactETA reduces the inference latency by more than 100 times compared to a state-of-the-art system, while maintains competing prediction accuracy.

KEYWORDS

estimated time of arrival, low-latency system, graph attention network, positional encoding

ACM Reference Format:
Kun Fu, Fanlin Meng, Jieping Ye, Zheng Wang. 2020. CompactETA: A Fast Inference System for Travel Time Prediction. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403386

1 INTRODUCTION

In the past few years, ride-hailing mobile apps, such as Uber, Lyft and Didi Chuxing, have been widely used. They effectively match the demand of riders and the capacity of drivers in real-time. The ride-sharing services provided by these online platforms are reforming the way of travel. They also help to improve the efficiency of the transportation system, which includes alleviating the increasing urban traffic congestion and reducing the carbon emission. All these efficient services provided by the platforms, such as vehicle dispatching and scheduling, are supported by high performance back end location based services (LBS).

(a) ETA example (b) ETA query complexity

Figure 1: (a) Estimated time of arrival refers to the estimated travel time between a pair of origin and destination along the given route. (b) Suppose we have 10 drivers and 20 riders with 5 candidate pickup positions in this simplified carpooling case, the order dispatch system has to request at least $10 \times 5^2 \times 20 \times 19 = 95000$ ETA queries to make the final decision.

The estimated time of arrival (ETA) is a core LBS in digital maps and navigation systems, which plays an important role in the ride-hailing scenario. In this paper, we define the ETA task as predicting the travel time from the origin to the destination along a given travel path. The path is represented as a sequence of road segments, referred as links. Figure 1 (a) presents a real example in which an accurate estimation of the travel time is made before the travel starts. An accurate ETA system is one of the basic infrastructures for the ride-hailing platforms to provide high quality transportation services to the customers. Nowadays, deep learning methods have achieved the state-of-the-art prediction result by solving a regression problem that outputs the estimated travel time. Recently, [28] and [25] introduce recurrent neural networks (RNN) to learn the travel time using the vehicle trajectory data. The recurrent network naturally adapts to the sequential data structure of the travel trajectory. However, the inference speed is not satisfactory in real applications, which leads to long latency and high resource consumption for current deep learning ETA solutions to be deployed in the production environment.

Unlike the conventional intelligence transportation system, the ride-hailing platforms make heavy use of ETA services. Figure 1 (b) gives an intuitive demonstration on the heavy duty of ETA service under a simplified carpooling scenario with two passengers

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403386

sharing the trip. Considering m drivers and n riders in this scenario, each rider having k candidate pickup choices, the decision process requests $O(mk^2n^2)$ ETA calls to make the optimal travel policy. With the fast growing scale of ride-hailing industry, there are surge demand of the real-time back end LBS, including ETA. For the world wide ride-hailing platforms, billions of ETA queries are triggered everyday by a wide range of applications such as vehicle dispatching, route planning and pricing. There could be a big operating cost. Thus in more and more scenarios, the inference performance of ETA module could become a bottleneck.

In this paper, we develop a novel ETA learning system named as CompactETA, which outputs an accurate travel time prediction within 100 microseconds in average. In order to obtain the high performance for inference, we use a very simple multi-layer perceptron (MLP) for online real-time inference. The simplified inference model requires properly learnt input features to retain satisfactory prediction accuracy. Thus we design an asynchronous feature representation learning module for CompactETA. This module takes all non-realtime raw features to learn spatiotemporal representations. In this module, we make explicit use of the road network structure information by applying a graph attention network on road network graph to learn spatial dependency between roads. We further adopt positional encoding, a technique widely used to replace the recurrent models such as in the Transformer [23], to encode the sequential information along the travel route and thus embed the temporal dependency in the representations. Finally, we take these asynchronously pre-computed representations concatenated with the real-time features as the input for the lightweight inference. We verify the performance gain of the proposed method with both offline experiments and online A/B testing. The results show that CompactETA reduces the inference latency by more than 100 times compared to the state-of-the-art system, while maintains competing prediction accuracy. After deployment, the proposed method saves significant amount of real-time computational resources.

Our contributions can be summarized as follows:

- We propose a novel ETA learning system named as CompactETA, which provides an accurate travel time prediction within 100 μs. To our best knowledge, this is the first work focusing on accelerating the inference speed of the ETA system and our proposed solution significantly outperforms existing ETA solutions by online inference speed.
- We apply graph attention network on road network to learn the spatiotemporal dependency between different roads and adopt positional encoding to encode the sequential information of travel route to avoid the recurrent network structure. The properly learnt feature representations enable us to use very simple model for inference.
- We deploy the proposed solution to the production environment and verify that the system preserves competing prediction accuracy but reduces the inference latency by more than 100 times compared to the state-of-the-art Wide-Deep-Recurrent (WDR) system.

2 APPROACH

We formulate the ETA problem in Section 2.1 and revisit the state-of-the-art deep learning baseline in Section 2.2. In Section 2.3, we present the proposed solution CompactETA. In CompactETA, we use a compact MLP model for real-time inference and shift the computation workload to an asynchronous representation learning module, which is a graph attention network followed by positional encoding. We design this graph attention network for road network graph in Section 2.4 and introduce detailed implementation of the positional encoding in 2.5.

2.1 Problem formulation

In the following paper, we formulate the travel time prediction as a regression problem. For an ETA query $q_i = \{d_i, \tau_i, \boldsymbol{p}_i\}$, where d_i is the driver ID, τ_i is the starting time and $\boldsymbol{p}_i = \{l_1, l_2, \cdots, l_{T_i}\}$ is a planned travel route with T_i road segments, our task is to construct sophisticated feature set $\mathcal{F}(q_i)$ and learn a model to map the input features to the travel time prediction $\tilde{y}_i$:

$$\mathcal{F}(q_i) \mapsto \tilde{y}_i. \tag{1}$$

In road network, we use link to refer to the physical road segment, which is characterized by a set of features. We denote the feature vector of link with ID = l as $\boldsymbol{x}_l$. The feature of all the links in the road network can be written as a matrix $\boldsymbol{L} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{\mathrm{N}}]$, where N is the total number of links. The link features include a group of static attributes such as length and number of lane, and a dynamic travel speed, which is periodically updated by an independent traffic service.

For query q_i, we construct a route feature set $\boldsymbol{X}_i$ by selecting a subset of the columns in $\boldsymbol{L}$ according to the travel path. It is represented as a variable-sized matrix $\boldsymbol{X}_i = [\boldsymbol{x}_{l_1}, \boldsymbol{x}_{l_2}, \cdots, \boldsymbol{x}_{l_{T_i}}]$, where T_i is the number of links along the travel path and varies for different trips. Each column x_{l_j} is a feature vector characterizing link l_j. The $\boldsymbol{X}_i$ integrates fine grained information for the travel route. We additionally construct a global feature vector $\boldsymbol{g}_i$ to put the features not related to route, such as weather type, driver ID and day of week. The $\boldsymbol{g}_i$ has fixed size and represents the deterministic factors for the travel. Thus, the raw input features for the ETA model are $\mathcal{F}(q_i) = (\boldsymbol{g}_i, \boldsymbol{X}_i)$. Figure 2 shows the difference between the two categories of features.

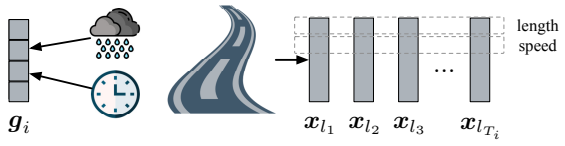

Figure 2: Two categories of features: the global feature g_i with fixed size and the route feature X_i whose size depends on the number of links along the travel path.

2.2 Baseline model

As focusing on product-level comparison, we choose the Wide-Deep-Recurrent (WDR) model [28] as our baseline model, which provides state-of-the-art prediction accuracy in both offline environment and online deployment. The discrete components of the input features are converted to distributed representations via embedding layers [19], and then appended to the continuous parts. We

set the embedding size for each component to 20. The intermediate result features generated from $\boldsymbol{g}_i$ and $\boldsymbol{X}_i$ are denoted as $\tilde{\boldsymbol{g}}_i$ and $\tilde{\boldsymbol{X}}_i$ respectively. Figure 3 shows the embedding and concatenating process of the feature transformation.

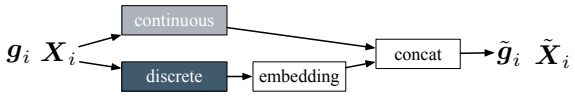

Figure 3: The feature processing workflow: we concatenate the raw continuous features with the embedding of the raw discrete features.

The WDR model has three parts of wide module, deep module and recurrent module. The wide network memorizes the patterns in the global features by using a cross product operation. The deep network uses two fully-connected layers to enhance the generalization ability. The recurrent network is a Long-Short Term Memory network (LSTM) [10] which sequentially reads $\tilde{\boldsymbol{X}}_i$ and summarizes the information into the last hidden state. The LSTM's initial states $\boldsymbol{c}_0$ and $\boldsymbol{h}_0$ are set to zero. At step t, the input gate $\boldsymbol{i}_t$, forget gate $\boldsymbol{f}_t$, output gate $\boldsymbol{o}_t$, modulated input $\boldsymbol{m}_t$, memory cell $\boldsymbol{c}_t$ and hidden state $\boldsymbol{h}_t$ are updated as follows:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_i[\tilde{\boldsymbol{x}}_{l_t};\boldsymbol{h}_{t-1}] + \boldsymbol{b}_i\right),\\
\boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_f[\tilde{\boldsymbol{x}}_{l_t};\boldsymbol{h}_{t-1}] + \boldsymbol{b}_f\right),\\
\boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_o[\tilde{\boldsymbol{x}}_{l_t};\boldsymbol{h}_{t-1}] + \boldsymbol{b}_o\right),\\
\boldsymbol{m}_t &= \tanh\left(\boldsymbol{W}_m[\tilde{\boldsymbol{x}}_{l_t};\boldsymbol{h}_{t-1}] + \boldsymbol{b}_m\right),\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{m}_t,\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{2}$$

where $\boldsymbol{W}_i, \boldsymbol{W}_f, \boldsymbol{W}_o$ and $\boldsymbol{W}_m$ are the weight matrices, $\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_o$ and $\boldsymbol{b}_m$ are the bias vectors, $\tilde{\boldsymbol{x}}_{l_t}$ is the feature vector for the t_{th} link in the path, $\sigma(.)$ is the Sigmoid function $\sigma(z) = 1/(1+e^{-z})$ and $\odot$ is the element-wise multiplication. We denote the outputs of wide and deep network as $\boldsymbol{h}_W$ and $\boldsymbol{h}_D$. The travel time prediction is given by a 3-layer MLP with the input of the concatenation of $\boldsymbol{h}_W, \boldsymbol{h}_D$ and the LSTM's last hidden state $\boldsymbol{h}_T$:

$$\tilde{y} = MLP\left(\boldsymbol{h}_W, \boldsymbol{h}_D, \boldsymbol{h}_T\right). \tag{3}$$

The overall structure of WDR model is given in Figure 4. The travel route generated from the ride-hailing platforms usually passes hundreds of links. For such long sequence, the recurrent module in WDR model becomes a heavy computational burden for online inference.

2.3 CompactETA

To obtain extremely fast online inference speed, we propose a novel ETA learning solution named as CompactETA. It is built up by two parts: a real-time predictor that uses compact model for online inference; and an asynchronous updater that learns high level representations from the input raw features. The overall structure is presented in Figure 5.

In the predictor, we use a 3-layer MLP to predict the travel time with the input of a linear combination of link representations and

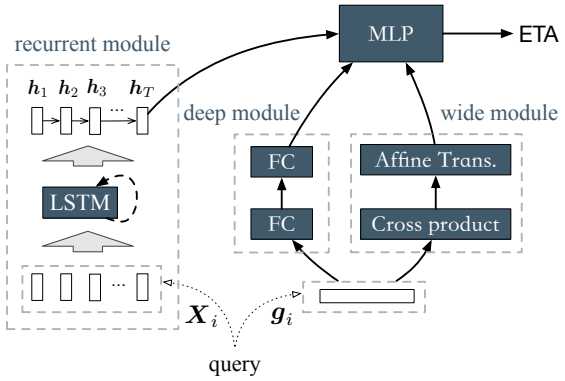

Figure 4: The structure of WDR model: the recurrent module is applied to the route feature while the wide and deep modules are applied to the global feature. The concatenation of the three modules' outputs is fed into a MLP to obtain the final prediction.

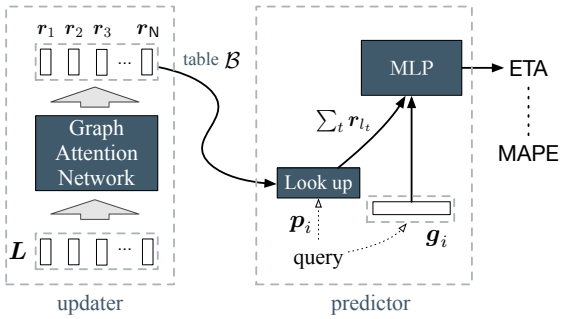

Figure 5: The structure of CompactETA. An updater learns high level representations for the links in map. When receiving an ETA query, a predictor looks link representations up in the table $\mathcal{B}$ and outputs the travel time prediction.

the global features:

$$\tilde{y} = MLP\left(\sum_{t=1}^{T} \boldsymbol{r}_{l_t}, \tilde{\boldsymbol{g}}\right), \tag{4}$$

where $\boldsymbol{r}_{l_t}$ is the representation vector of link l_t and $\tilde{\boldsymbol{g}}$ is the transformed global features. The $\boldsymbol{r}_{l_t}$ is obtained by searching with the key l_t through the pre-computed table $\mathcal{B} = \{\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_N\}$, which stores the representations for all the links in the road network.

Suppose we have a simple predictor with size s for all input and hidden layers, the inference of CompactETA costs $O(s^2)$ multiplications and $O(T)$ table lookup operations. However, the inference of the WDR model at the same scale costs at least $O(Ts^2)$ multiplications only in the recurrent module according to Equation 2. This comparison shows significant inference acceleration from CompactETA.

The updater is used to learn the lookup table of link representations. The properly learnt feature representations ensure CompactETA to retain satisfactory prediction accuracy. In this updater,

we apply a graph attention network to learn the corresponding representations which capture the spatiotemporal dependency in the road network. The effectiveness of modeling road network as graph has been studied in small or medium scale traffic forecasting problems [18, 31]. However, it is not affordable to directly apply standard spectral operation on large-scale graphs with millions of nodes as in our problem. Motivated by [24], we introduce attention mechanism to the road network graph. Compared to spectral graph convolution methods like [12], the attention mechanism is parameter-efficient for large graphs. Moreover, its computation can be further accelerated by parallelization.

As introduced in Section 2.1, the link related features keep unchanged before traffic updates, whose period is $\Delta = 120$ seconds in our setting. Thus the link representations $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_N\}$ can be saved in the table $\mathcal{B}$ and reused for different queries within this period. And $\mathcal{B}$ is updated by the updater in every Δ seconds.

The asynchronous computation only occurs at inference stage. The whole CompactETA model is trained end-to-end. The parameters of embedding layers, the updater and the predictor are trained jointly by minimizing the overall mean absolute percentage error (MAPE) with Adam [11] optimizer:

$$\min_{\theta} \sum_{i} \left| \frac{y_i - \tilde{y}_i}{y_i} \right|, \tag{5}$$

where θ represents the model parameters, y_i is the ground-truth travel time of i_{th} query and $\tilde{y}_i$ is the predicted travel time.

2.4 Graph attention network

We build the road network graph to capture the road network topology by considering the links as nodes and their connectivity as edges. We use a one-hot vector γ_{ij} to indicate the connection status between link l_i and l_j:

$$\gamma_{ij} = \begin{cases} [1,0,0], & \text{if } l_i \rightarrow l_j; \\ [0,1,0], & \text{if } l_i \leftarrow l_j; \\ [0,0,1], & \text{if } l_i \leftrightarrow l_j, \end{cases} \tag{6}$$

where $l_i \rightarrow l_j$ means that a vehicle on l_i can reach l_j without passing any other link, $l_i \leftarrow l_j$ and $l_i \leftrightarrow l_j$ can be explained in the same way according to the direction of the arrow. We define $\gamma_{ii} = [0,0,1]$ for self connection.

We build our graph attention network by stacking 3 graph attention blocks. Let $k = 1, 2, 3$ be the block index and $\boldsymbol{u}_i^{(k)}$ be the output for node i in the k_{th} block. For block k, we first compute the correlation score between each pair of connected links:

$$\boldsymbol{e}_{ij}^{(k)} = Att^{(k)}\left(\boldsymbol{u}_i^{(k-1)}, \boldsymbol{u}_j^{(k-1)}, \gamma_{ij}\right), \tag{7}$$

where $Att^{(k)}$ stands for a 1-hidden-layer MLP. The score is determined by the features and connection status of the links. Then we convert link score vector into a distribution over neighbors by softmax:

$$\boldsymbol{\alpha}_{ij}^{(k)} \propto exp(\boldsymbol{e}_{ij}^{(k)}), j \in \mathcal{N}_i, \tag{8}$$

where $\mathcal{N}_i$ includes l_i and its nearest neighbors. We update the state of l_i using the weighted sum on $\mathcal{N}_i$:

$$\boldsymbol{v}_i^{(k)} = \sum\nolimits_{j \in \mathcal{N}_i} \boldsymbol{\alpha}_{ij}^{(k)} \cdot \boldsymbol{u}_j^{(k-1)}. \tag{9}$$

The attention blocks are stacked in the same way as residual network:

$$\boldsymbol{u}_i^{(k)} = \boldsymbol{u}_i^{(k-1)} + relu\left(\boldsymbol{W}_2^{(k)} \cdot relu\left(\boldsymbol{W}_1^{(k)} \boldsymbol{v}_i^{(k)} + \boldsymbol{b}_1^{(k)}\right) + \boldsymbol{b}_2^{(k)}\right), \tag{10}$$

where $\boldsymbol{W}_1^{(k)}$, $\boldsymbol{W}_2^{(k)}$, $\boldsymbol{b}_1^{(k)}$ and $\boldsymbol{b}_2^{(k)}$ are the parameters of the fully connected layers. Figure 6 explains the computation flow inside a graph attention block.

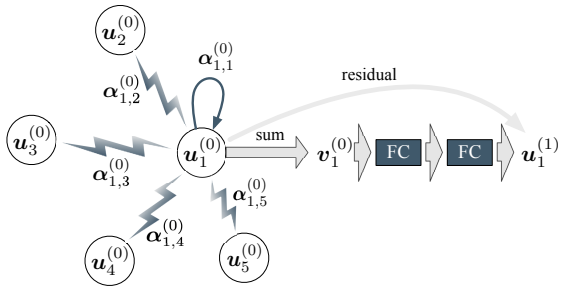

Figure 6: Graph attention block with $k = 1$: the state of link l_1 is updated by a residual connection skipping over the block of nearest neighborhood attention followed by two fully connected layers

We introduce an affine transformation before the first attention block to match the input and output sizes for the residual connection. It initialize the states as $\boldsymbol{u}_i^{(0)} = \boldsymbol{W}_{af}\tilde{\boldsymbol{x}}_{l_i} + \boldsymbol{b}_{af},\ i = 1, 2, \cdots, N$, where $\tilde{\boldsymbol{x}}_{l_i}$ is the link feature. $\boldsymbol{W}_{af}$ and $\boldsymbol{b}_{af}$ are the transformation parameters. For the k_{th} block, $\boldsymbol{u}_i^{(k)}$ contains information from as far as link l_i's k-hop neighbors. The increasing receptive field makes higher level neurons see larger range of the road network graph, which is similarly as in the convolutional neural network (CNN) [14]. Larger receptive field usually results high prediction accuracy. However, the computational cost grows fast as we stack more graph attention blocks. In practice, we found balance by stacking 3 blocks. We use the output of attention network as the link representation, namely, $\boldsymbol{r}_i = \boldsymbol{u}_i^{(3)}$.

2.5 Positional encoding

The road segments along each travel route have clear sequential structure. This is not neglectable for training a high accurate travel time prediction model. However, we cannot afford recurrent network to build a fast inference system. Thus we adopt positional encoding technique to encode the sequential information. Positional encoding is initially proposed for natural language processing task in the design of Transformer [23]. For our problem, we generate a series of cosinusoids:

$$PE(pos, dim) = \cos\left(\frac{pos}{10000^{dim/d_{rep}}}\right), \tag{11}$$

where pos is the link's position in a travel path (from 1 to T), dim is the dimension index of link representation and d_{rep} is the size of link representation. We denote $\boldsymbol{PE}_t = PE(t, :)$ as the positional encoding vector for position t.

In Transformer, the positional encoding vector is added to the input word. However, addition is meaningless in our case, because $\sum_t(\boldsymbol{r}_{l_t} + \boldsymbol{PE}_t) = \sum_t \boldsymbol{r}_{l_t} + \sum_t \boldsymbol{PE}_t$ just adds a constant term in the original form and ignores the link order as well. We propose a more aggressive way by multiplying the link representation by the positional encoding:

$$\sum_t (1 + \beta \cdot \boldsymbol{PE}_t) \cdot \boldsymbol{r}_{l_t}, \tag{12}$$

where β is a hyper-parameter to control the multiplication. In online environment, the cosinusoids can be computed in advance, and the element-wise multiplication increases the inference latency by a small margin.

After introducing the graph attention network and positional encoding, we present the complete working flow for CompactETA in Figure 7. The renewal of the traffic condition triggers the updater to recompute the link representations. Then it pushes the new representations to the predictor. After receiving an ETA query at the inference stage, the predictor selects the representations in the table $\mathcal{B}$ for the links in the planned route, then applies positional encoding and finally outputs the travel time prediction inferred by the MLP.

3 RELATED WORK

The rapid growth of the travel data produced on ride-hailing platforms makes data-driven methods an effective way to accurately predict ETA (a.k.a travel time). These methods either build rules or apply machine learning models to extract the spatial and temporal patterns in the historical trajectories collected from floating-cars. For example, [27] represents the travel time for each road by a certain driver at a specific time slot using a 3-D tensor, and learns the travel time for each element in the tensor via tensor decomposition. [5] predicts the future traffic pattern along the candidate travel path using a feed-forward neural network and matches the traffic pattern among historical trips to estimate the travel time. Similar works have been done to predict ETA by directly using similar historical trips. [26] finds neighboring trips with nearby origin and destination, and makes ETA prediction by adjusting the average of the neighbor's travel time based on traffic periodicity. [22] introduces online learning to travel time prediction problem based on extended Kalman filter, which alleviates the delayed-label problem that trip could be finished after a long time. [1] and [16] predict ETA distribution instead of a single value. [1] directly learns the probabilistic density function and [16] learns a deep generative model. ETA model could also be learnt by regression tasks. [30] adopts support vector regression, [4, 7, 28] introduce recurrent neural network and [17] uses multi-layer perceptron to learn the nonlinear regression function. [25] learns a deep neural network on GPS sequence, which focuses on offline analysis but is unavailable for practical online inference, as the ETA prediction should be provided before travel starts in most cases. At that moment, GPS sequence for the travel is not available yet. [6] transforms the map and traffic features into images and predicts ETA using vision models like CNN. [13, 15, 21, 29] focus on bus ETA and propose models more suitable for bus trajectory data.

The path based model (e.g [28]), whose input includes a planned travel route, usually outperforms the path-free model (e.g [17]), whose input only includes the origin and the destination locations but has no information about the travel route. In recent studies, recurrent neural network is presented as a popular ingredients in the state-of-the-art models for its advantage of capturing the sequential information along the travel path, although it is computational expensive. Our proposed CompactETA removes the recurrent network structure to significantly reduce the computational consumption without losing prediction accuracy.

The graph convolution method gains rapid progress in recent years. The convolution has been generalized from fixed-grid to the graph domain. The graph convolution is initially defined by [2] in Fourier domain based on eigendecompostition of the graph Laplacian. [9] introduces a spatially localized version of the spectral filter and [3] proposes to avoid computing the eigenvectors by a Chebyshev expansion of the graph Laplacian. [12] further simplifies the computation by constraining the filter to be applied to only the nearest neighbors. All these methods are categorized as spectral methods. The graph attention network [24] is developed along another research branch [8, 20] — non-spectral methods which directly define operations on graph structures. The attention operation is parameter-efficient for large graphs and its local computation can be parallelized by taking advantages of GPUs.

4 OFFLINE EXPERIMENT

We empirically evaluate the proposed CompactETA on large-scale historical data, and then deploy it to production environment to test the online results. The offline experiments are introduced in this section and the online testings are presented in Section 5.

4.1 Data

The floating-car data are collected on DiDi platform from 3 cities in China: Beijing, Suzhou and Shenyang. According to the driver's working status, the data are divided into two types — pickup data and trip data. A pickup sample starts when the driver responds to a rider's request and ends when the rider is picked up. A trip sample starts when a rider gets on board and ends when the rider arrives the destination. The GPS trajectory is matched to the road network to obtain the corresponding sequence of traveled roads. We remove the outliers which have very short travel time (< 60s) or extremely high average speed (> 120km/h). The collection time window is from Jun 3, 2019 to Sep 1, 2019 and the data are split to training set, validation set and test set, which have 70, 7 and 14 days of data, respectively. Table 1 summarizes the numbers of sample, driver and link.

Table 1: Statistics of the datasets.

	Beijing		Suzhou		Shenyang	
	pickup	trip	pickup	trip	pickup	trip
train	50.5M	50.9M	17.3M	19.4M	5.1M	5.3M
val	5.1M	5.1M	1.8M	2.0M	0.4M	0.4M
test	10.1M	10.0M	3.5M	3.9M	1.0M	1.1M
driver	687k	687k	401k	401k	145k	145k
link	2.1M	2.1M	1.4M	1.4M	0.9M	0.9M

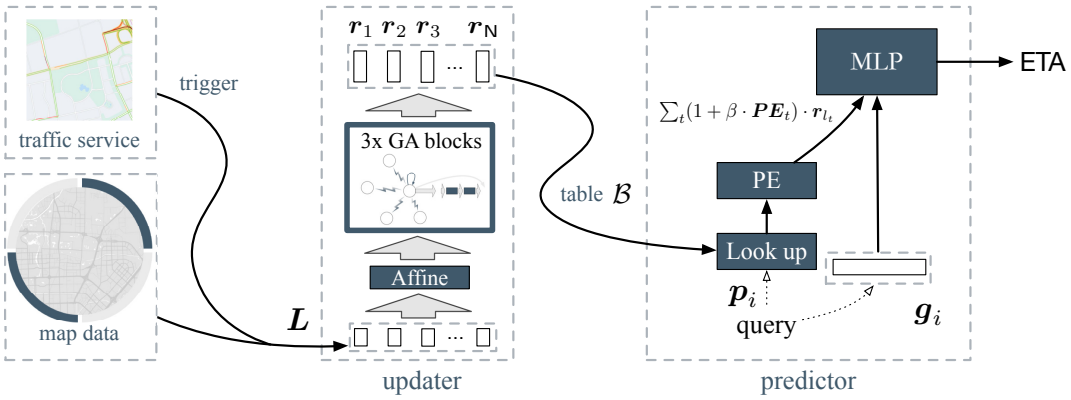

Figure 7: The workflow of CompactETA: the traffic renewal triggers the updater, which recomputes the link representations and pushes them to the predictor. The predictor receives ETA queries and provides the prediction result inferred by the simple MLP model.

4.2 Competing methods

In the offline experiment, we compare our CompactETA with two representative methods — RouteETA and the WDR model. For a more comprehensive study, we also add three variants of CompactETA to validate the detailed benefit from different parts of the model.

RouteETA is a rule-based solution which is widely used in map service and navigation systems. It estimates the travel time along a given path by summing up the expected travel time of each link and the expected delay time at each intersection. An expected link travel time is obtained by dividing its length by the average travel speed within the time window of traveling. This solution provides fast ETA inference due to its simplicity. However, the prediction accuracy is not satisfactory.

WDR model is a deep learning solution which combines wide, deep and recurrent neural network together and achieves the state-of-the-art prediction accuracy. It provides accurate travel time prediction but consumes massive computational resources at inference stage. In the experiment, we set the hidden sizes in the recurrent and deep modules to 128.

MLP-ETA is a variant of CompactETA without graph attention and positional encoding. For fair comparison, it uses a 6-layer MLP with residual connection every 2 layers to learn the link representation. It thus has comparable model depth to the CompactETA. Moreover, we evaluate **MLP-ETA-GA** and **MLP-ETA-PE** which introduce the graph attention and positional encoding to MLP-ETA, respectively. These 3 variants are used for ablation study and their hidden sizes are all set to 128. For all the models in our experiment, we set $\beta = 1.0$ if positional encoding is applied, use a learning rate of 0.0001 and a batch size of 256, and adjust the Adam optimizer of embedding parameters to a variant SparseAdam [1].

4.3 Results

We implement the models in Python using Caffe2 framework and accelerate the training on Nvidia K40 GPUs. Using one GPU card, the training process of CompactETA finishes within 10 hours for pickup data and 20 hours for trip data for each city.

[1]Refer to https://pytorch.org/docs/stable/optim.html#torch.optim.SparseAdam

Table 2: Comparing results in terms of MAPE for Beijing, Suzhou and Shenyang (%).

Pickup	Beijing	Suzhou	Shenyang
WDR	18.37	18.25	19.12
RouteETA	25.54 (+7.17)	26.20 (+7.95)	24.93 (+5.81)
MLP-ETA	19.25 (+0.88)	19.02 (+0.77)	19.46 (+0.34)
MLP-ETA-PE	19.02 (+0.65)	18.86 (+0.61)	19.37 (+0.25)
MLP-ETA-GA	18.59 (+0.22)	18.52 (+0.27)	19.28 (+0.16)
CompactETA	18.58 (+0.21)	18.45 (+0.20)	19.05 (**-0.07**)
Trip	**Beijing**	**Suzhou**	**Shenyang**
WDR	11.02	10.89	11.19
RouteETA	16.28 (+5.26)	15.78 (+4.89)	16.30 (+5.11)
MLP-ETA	11.68 (+0.66)	11.54 (+0.65)	11.80 (+0.61)
MLP-ETA-PE	11.66 (+0.64)	11.55 (+0.66)	11.76 (+0.57)
MLP-ETA-GA	11.20 (+0.18)	11.08 (+0.19)	11.36 (+0.17)
CompactETA	11.19 (+0.17)	11.07 (+0.18)	11.35 (+0.16)

Table 2 shows the comparing result on Beijing, Suzhou and Shenyang in terms of the mean absolute percentage errors. The numbers in the brackets indicate the MAPE differences compared to the WDR model (lower the better). The prediction accuracy gaps between our proposed model and the state-of-the-art WDR model are around 0.2% on both pickup and trip data (CompactETA v.s WDR). Note that on pickup data of Shenyang, CompactETA even outperforms the WDR model by 0.07%. For a 20 minutes order, the 0.2% gap means that the prediction from CompactETA is expected to have additional 2.4 seconds error compared to the WDR model. Considering more than 100 times speeding up on inference, CompactETA is more competitive for real applications.

The graph attention technique obtains 0.2% to 0.7% performance boost over the simple MLP model which neglects link dependency

(MLP-ETA-GA v.s MLP-ETA). Larger city benefits more from graph attention than smaller city on pickup data, and all the cities obtain similar gains around 0.4% on trip data. Applying positional encoding can further decrease the MAPEs by up to 0.24% on pickup data (CompactETA v.s MLP-ETA-GA). However, only slight improvement is observed on trip data. The performance of another popularly used solution in industry RouteETA is much worse than the deep learning models in most cases.

4.4 Visualization of link representation

In this experiment, we extract the learnt link representations generated by the graph attention network and project them into a 2-D space by Principal Component Analysis (PCA). We randomly select 1,000 samples from the validation sets to visualize the PCA results for each city and each data type in Figure 8. We can clearly observe that the relative positions of the link representation are highly correlated with the link travel time. For example in Figure 8 (1a), the link representation gradually moves from the left top corner to the right bottom corner as the travel time increases.

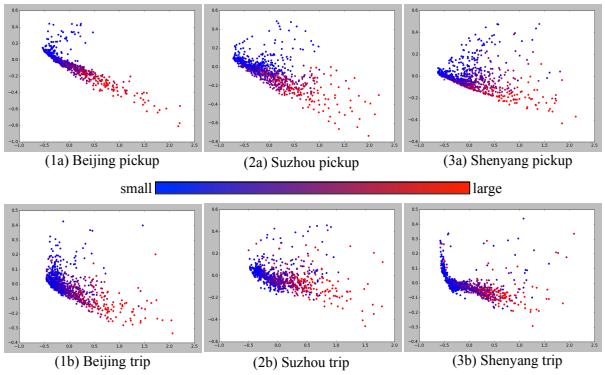

Figure 8: PCA result of link representations for 1000 random selected samples in 2-D space. The point is colored according to its travel time by a linear interpolation between blue and red in color space.

We design another experiment to visualize the semantic of the link representations. We randomly select a path from Suzhou trip data and compute the cumulative sums of the first 1 to 5 link representations. These 5 result vectors are projected to the same 2-D space in Figure 8 (2b). Figure 9 shows the results and shows that the cumulative sum vector moves from the small travel time area to the large travel time area as the route becomes longer with more representations being accumulated. It demonstrates that accumulating link representations has similar effect to directly accumulating the link travel time.

5 ONLINE EVALUATION

In this section, we present the online A/B testing for CompactETA and the WDR model. We also report the statistics of the service performance to compare the consumption of computational resources.

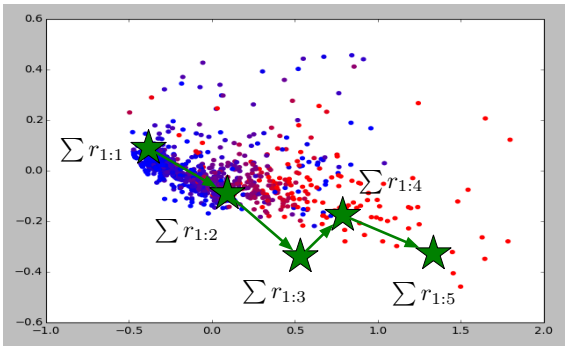

Figure 9: The trajectory of cumulative sums of link representations along the travel path. $\sum r_{1:t}$ is the sum for the first t links and the green stars mark the first 1 to 5 sums.

5.1 Architecture

We deploy the predictor to CPU clusters. In the implementation, we use a C++ linear algebra library Eigen with Math Kernel Library backend. The updater is deployed to GPU clusters. Once the input traffic information changes, the updater inferences the link representations for the entire map network and sends the results to the predictor via remote procedure call. Since the updating process is asynchronous to the online inference, it can be efficiently executed in batch mode. We choose Caffe2's C++ interface, which has a lightweight overhead, to implement the graph attention network. In our experiment, one updater is able to support at least 10 CPU servers running the predictor.

5.2 A/B testing

A/B testing is a randomized experiment to compare the effectiveness of two strategies. It splits the queries into control group (A) and treatment group (B) and apply different strategies to them. Statistical hypothesis testing on the mean of a certain metric is then taken to determine which of the strategy is more effective. The A/B testing is conducted in the production environment with real user requests, and it is more convincing than offline experiments to evaluate the impact of the competing solutions.

We deploy the A/B testing on Suzhou, a major city in the east China. In our A/B testing, we estimate the travel time for a query and compute the metrics until we can collect the ground-truth travel time after the order finishes. Though we may receive a large number of queries in the ETA service, only a small portion of them become valid samples with a closed deal for the order request and the ground-truth travel time available. We use the CompactETA model if a query belongs to the treatment group, and apply baseline model to the control group. The treatment and control groups are split by a 3-hour rotation strategy. Specifically, we switch the same group people as control or treatment group every 3 hours. This strategy is more reasonable and convincing than random splitting for two-sided market such as in ride-hailing scenario, which is different from conventional web services.

As the peak hours are not uniformly spread over the groups, we further switch the phase of rotation every day — for one day,

testing starts with control group and the following day starts with treatment group. We also let the A/B testing duration as a multiple of 14 days, to alleviate the impact of the supply and demand condition in different days of the week. Even though, it is still difficult to ensure absolute fairness. Some random factors such as weather change or temporary traffic control can hurt the fairness. Thus, we first run a two weeks A/A testing which has both treatment and control groups using the baseline model. The results are presented in the top half of Table 3,

Table 3: A/A and A/B testing results for Suzhou.

A/A	Control	Treatment	Rel. diff.	p-value
sample	461k	458k	-0.65%	-
pickup MAPE	32.208%	32.213%	+0.02%	0.977
pickup badcase	2.103%	2.085%	-0.86%	0.594
trip MAPE	19.651%	20.125%	+2.41%	0.031
trip badcase	2.547%	2.493%	-2.12%	0.249
dissatisfaction	1.504%	1.461%	-2.86%	0.182
A/B	**Control**	**Treatment**	**Rel. diff.**	**p-value**
sample	445k	436k	-2.02%	-
pickup MAPE	33.387%	34.113%	+2.17%	3×10^{-7}
pickup badcase	2.020%	1.843%	**-8.76%**	2×10^{-8}
trip MAPE	19.095%	18.685%	-2.15%	5×10^{-4}
trip badcase	1.882%	1.838%	-2.34%	0.324
dissatisfaction	1.408%	1.370%	-2.70%	0.179

The results of the two weeks A/B testing are listed in the bottom half of Table 3. We report the MAPE as well as the badcase rate. A query is treated as a badcase when $|y - \tilde{y}| > 5$min for pickup data and $|y - \tilde{y}| > 10$min for trip data. We additionally report the user dissatisfaction rate as an indicator of the user experience. Most of the metric differences between the CompactETA and the WDR model are comparable to the natural noise in A/A testing — except that there is a significant improvement of 8.76% on pickup badcase rate. The improvements of badcase rate is mainly due to the explicit summation of the links' representations. In WDR, the summation of link representations is implicitly conducted in LSTM, which has less numerical ability to adapt the wide range of ETA outputs (from one to thousands of seconds). These results validate that CompactETA achieves competing results as the state-of-the-art model. Figure 10 shows the daily trends of MAPE in the A/B testing.

Note that the values for online metrics are usually not as good as those in offline experiments. For example, a user may choose a route other than the planned one, which is used in prediction, or even change destination in the middle of the trip.

5.3 Service quality

We use these experiments to verify that CompactETA saves a lot of computational resources compared to the state-of-the-art deep learning methods even in production environment. We compare the key indicators of service quality between CompactETA and the WDR model in deployment.

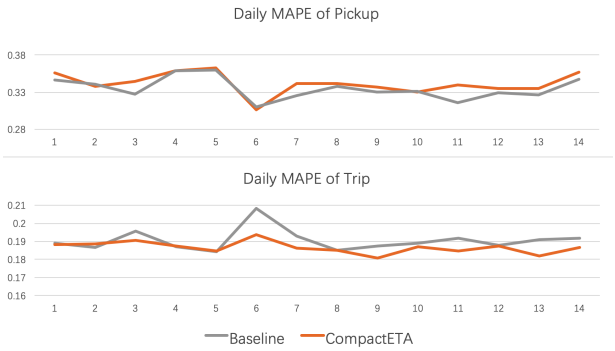

Figure 10: The daily MAPE trends for pickup data (top) and trip data (bottom) in the A/B testing. The gray and orange curves stand for the control and treatment groups, respectively.

Table 4: Comparing result of service quality.

	WDR		CompactETA	
Concurrency	QPS	Avg. Latency	QPS	Avg. Latency
1	0.23k	4.3 ms	33.0k	0.030 ms
5	1.16k	4.3 ms	65.3k	0.031 ms
10	2.32k	4.3 ms	294k	0.034 ms
20	4.47k	4.5 ms	555k	0.036 ms
30	5.49k	5.5 ms	722k	0.042 ms
40	6.23k	6.4 ms	769k	0.052 ms
50	6.69k	7.5 ms	765k	0.065 ms
70	6.82k	10.3 ms	934k	0.075 ms
100	6.80k	14.7 ms	918k	0.109 ms
150	6.89k	21.8 ms	892k	0.168 ms
200	6.96k	28.8 ms	807k	0.248 ms

In this experiment, we call the ETA service with simulated queries constructed from historical trips. The throughput of requests is a hyper-parameter and can be adjusted to evaluate the service performance under different stress levels. We focus on two key statistics: (1) query per second (QPS) indicating the throughput capacity of a service, and (2) latency which is the time between making a request and the completion of the response. In large-scale cloud, the network delay is non-negligible. Thus we pack 60 queries in each request to improve efficiency[2].

The services are deployed to a cloud powered by Intel Xeon E5-2670 2.30GHz server with 48 CPU cores and 128GB memory. Table 4 lists the QPS and average latency of CompactETA and the WDR service under different stress levels (controlled by the request concurrency). More detailed statistics are visualized in Figure 11. The results show that CompactETA is capable of handling 100 times more queries per second and reduces the average latency by more than 100 times compared to the WDR system. It means that the ETA service workload provided by 1,000 machines using WDR model

[2]The QPS and average latency are computed based on queries, not requests.

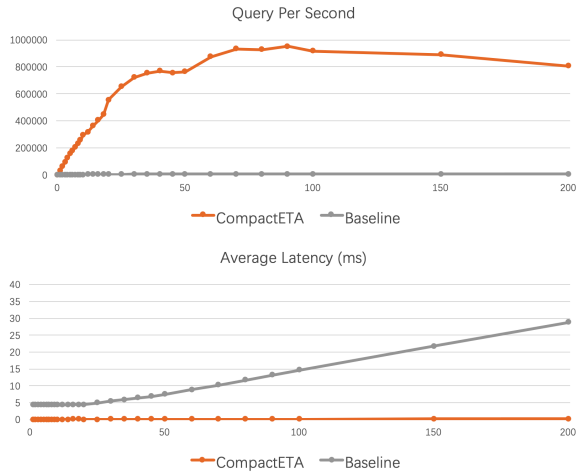

Figure 11: The QPS and average latency under different stress levels (X-axis stands for request concurrency). CompactETA achieves much better QPS and latency performance than the WDR system.

can be fulfilled by 10 machines[3] using CompactETA. For large ride-hailing platforms, this can significantly save the computational resources.

6 CONCLUSION

In this paper, we introduce a novel ETA system CompactETA, which provides a precise travel time prediction within 100 μs. In this solution, we use a compact model for real-time inference. The inference model use high level link representations as input feature, which is learnt on road network graph by a graph attention network equipped with positional encoding. These representations capture the spatiotemporal dependency between roads, and further encode the sequential information of the travel route. Large-scale offline experiments and online A/B testing validate that CompactETA boosts the inference speed by more than 100 times compared to the state-of-the-art WDR model, while maintains competing prediction accuracy. In real applications, CompactETA saves considerable operation cost for ride-hailing platforms. The proposed solution can also be adopted to build general high performance machine learning systems with proper graph representations, such as the real time machine learning modules in intelligence transportation or network analysis.

REFERENCES

[1] Mohammad Asghari, Tobias Emrich, Ugur Demiryurek, and Cyrus Shahabi. 2015. Probabilistic Estimation of Link Travel Times in Dynamic Road Networks. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems (NeurIPS)*. 3844–3852.

[4] Yanjie Duan, Yisheng Lv, and Fei-Yue Wang. 2016. Travel time prediction with LSTM neural network. In *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC '16)*.

[3]Plus an extra GPU machine

[5] C. de Fabritiis, R. Ragona, and G. Valenti. 2008. Traffic Estimation And Prediction Based On Real Time Floating Car Data. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*.

[6] Tao-yang Fu and Wang-Chien Lee. 2019. DeepIST: Deep Image-based Spatio-Temporal Network for Travel Time Estimation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 69–78.

[7] Ruipeng Gao, Xiaoyu Guo, Fuyong Sun, Lin Dai, Jiayan Zhu, Chenxi Hu, and Haibo Li. 2019. Aggressive driving saves more time? multi-task learning for customized travel time estimation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19). AAAI Press*. 1689–1696.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*.

[9] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[12] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[13] M. Kormáksson, L. Barbosa, M. R. Vieira, and B. Zadrozny. 2014. Bus Travel Time Predictions Using Additive Models. In *IEEE International Conference on Data Mining (ICDM '14)*.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[15] Wang-Chien Lee, Weiping Si, Ling-Jyh Chen, and Meng Chang Chen. 2012. HTTP: A New Framework for Bus Travel Time Prediction Based on Historical Trajectories. In *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.

[16] Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. 2019. Learning Travel Time Distributions with Deep Generative Model. In *The World Wide Web Conference*. ACM, 1017–1027.

[17] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1695–1704.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3111–3119.

[20] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5115–5124.

[21] Charul Paliwal and Pravesh Biyani. 2019. To each route its own ETA: A generative modeling framework for ETA prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 3076–3081.

[22] JWC Van Lint. 2008. Online learning solutions for freeway travel time prediction. *IEEE Transactions on Intelligent Transportation Systems* 9, 1 (2008), 38–47.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*. 5998–6008.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[25] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? estimating travel time based on deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[26] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A simple baseline for travel time estimation using large-scale trip data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.

[27] 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 (KDD '14)*.

[28] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 858–866.

[29] Jiancheng Weng, Chang Wang, Hainan Huang, Yueyue Wang, and Ledian Zhang. 2016. Real-time bus travel speed estimation model based on bus GPS data. *Advances in Mechanical Engineering* 8, 11 (2016), 1687814016678162.

[30] Chun-Hsin Wu, Jan-Ming Ho, and D. T. Lee. 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems* 5, 4 (Dec 2004), 276–281.

[31] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *International Joint Conference on Artificial Intelligence (IJCAI)*.