Learning to Effectively Estimate the Travel Time for Fastest Route Recommendation

Ning Wu, Jingyuan Wang*
Beijing Advanced Innovation Center for BDBC, School of Computer Science and Engineering, Beihang University, Beijing, China
{wuning,jywang}@buaa.edu.cn

Wayne Xin Zhao*
School of Information, Renmin University of China
Beijing, China
batmanfly@gmail.com

Yang Jin
MOE Engineering Research Center of ACAT, School of Computer Science and Engineering, Beihang University, Beijing, China
jy0205@buaa.edu.cn

ABSTRACT

Fastest Route Recommendation (FRR) aims to find the fastest path in response to user's queries in a large complex road network. Early studies cast the FRR task as a pathfinding problem on graphs and adopt heuristic algorithms as the major solution due to the efficiency and robustness. A major problem of heuristic algorithms is that the heuristic function is usually empirically set with simple methods, which is difficult to model other useful factors. In this paper, we extend the classic A^* algorithm for the FRR task by modeling complex traffic information with neural networks. Specially, we identify an important factor that is important to improve the FRR task, *i.e.,* the estimation of travel time. For this purpose, we first develop a module for predicting the time-varying traffic speed for a road segment, which is the foundation for estimating the travel time. Conditioned on this module, we further design another module to estimate the fastest travel time between two locations connected by routes. We adopt neural networks to implement both modules for enabling the capacity of modeling complex traffic characteristics and dynamics. In this way, the original two cost functions of A^* algorithm have been set in a more principled way with neural networks. To our knowledge, we are the first to use neural networks for improving A^* algorithm in the FRR task. It elegantly combines the merits of A^* algorithm and the powerful modeling capacities of neural networks for the FRR task. Extensive results on the three real-world datasets have shown the effectiveness and robustness of the proposed model.

CCS CONCEPTS

• **Information systems → Location based services**; **Recommender systems**; Geographic information systems; Global positioning systems.

KEYWORDS

Fastest route planning; Traffic speed prediction; Heuristic search

*Corresponding author.

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

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357907

ACM Reference Format:
Ning Wu, Jingyuan Wang, Wayne Xin Zhao, and Yang Jin. 2019. Learning to Effectively Estimate the Travel Time for Fastest Route Recommendation. 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/10.1145/3357384.3357907

1 INTRODUCTION

Due to global urbanization, people have increasing demands for accurate and real-time navigational information through the complex road network. To better facilitate the travel of users, *fastest route recommendation (FRR)* has become an important task in urban computing. Given the road network and corresponding traffic condition information, FRR aims to generate a or several fastest route suggestions on instant queries about the path planing from a source to a destination [4, 6, 9, 27–29]. It is challenging to perform effective pathfinding in a large road network, since it requires a comprehensive consideration of real-time and historical traffic data according to road network information.

Early studies cast the FRR task as a pathfinding problem on graphs and solve this task via heuristic search [6, 9, 16], *e.g.,* Dijkstra and A^* algorithms. Due to its efficiency and robustness, heuristic search has become one of the main solutions for the FRR task. A key point of these work is to develop an effective heuristic function. Take A^* algorithm as an example. It aims to find a path to the given destination node resulting in the smallest cost. A^* evaluates a candidate node n based on a cost function $f(n)$, which will be decomposed into two parts, namely the observable cost from the source to the evaluation node $g(n)$ and the estimated cost from the evaluation node to the destination $h(n)$. $h(n)$ is often called *heuristic function*, which needs to be set according to specific tasks. The effectiveness of $h(n)$ directly determines the final performance of A^* algorithm. With suitable heuristics, heuristic algorithms can substantially reduce the search space and obtain high-quality responses. However, traditional heuristic search algorithms usually require to set the heuristic function manually or using simple statistical methods [6, 9, 9, 16, 22], which limits the flexibility and extensibility. It will be difficult to integrate other influencing factors or useful components in heuristic search algorithms.

As shown in previous studies [4, 6, 9, 27–29], an important factor to improve the route recommendation performance is to accurately acquire the travel time between two locations in a road network. Travel time is particularly useful for the route planing task. For example, in A^* algorithm, the essence of $h(n)$ function is to estimate an effective cost (*i.e.,* travel time cost for the FRR task) from the current

evaluation node to the destination node. However, the travel time is usually unavailable and difficult to be predicted, since the traffic condition is varying over time. Therefore, an ideal FRR method should have the capacity of predicting the travel time according to real-time traffic condition, and further make more reliable recommendation based on its predictions. Following this idea, if we were able to integrate heuristic search algorithms with travel time prediction models, can we develop a more effective FRR method?

To implement such a method, there are at least three challenges to address. First, heuristic search algorithms are general algorithm framework, and it is difficult to integrate predictive modules into the framework. Second, supposing that we can obtain the estimation related to travel time from the predictive modules, it is not clear how to fully utilize such useful information to guide the search process. Third, although travel time prediction has been extensively studied in the literature [4, 6, 9, 27–29], they have seldom been studied in the FRR task. It needs to develop an effective predictive module for the FRR solution as a whole.

To address these difficulties, we propose to integrate A^* algorithm with neural network based on travel time estimation for solving the FRR task. Given the fact that a route consists of a sequence of road segments, we identify a primary technical difficulty for estimating the travel time through a route, *i.e.,* the prediction of the traffic speed for a road segment. For a road segment, we do not assume a static speed, since there would be significant speed variation at different time under time-varying traffic condition. For this purpose, we first develop a module for predicting the time-varying traffic speed for a road segment, which can be used to compute the time for passing a road segment. For effectively capturing the complex traffic characteristics, we jointly characterize short-term trend, long-term temporal patterns and spatial influence. To further estimate the travel time, we develop another module to estimate the fastest travel time between two locations connected by routes. We adopt neural networks to implement both modules for enabling the capacity of modeling complex traffic characteristics and dynamics.

After obtaining the predicted travel speed of a road segment, we can set $g(\cdot)$ by adding the existing time cost with the time through a candidate road segment towards the evaluation node. Furthermore, we can set $h(\cdot)$ with the cost from a candidate node to the destination using the estimated travel time between them. In this way, the original two cost functions have been set in a more principle way with neural networks. After that, we perform a standard search procedure of A^*. It elegantly combines the merits of A^* algorithms and the powerful modeling capacities of neural networks for the FRR task. Although the heuristics are learned from neural networks, we have shown that it is able to achieve a high probability to identify the optimal route under some reasonable assumptions.

To the best of our knowledge, we are the first to use neural networks for improving A^* algorithm in the FRR task. Our approach is able to automatically learn the cost functions without handcrafting heuristics. It is able to effectively predict the arrival time through a road segment and a distant pair of locations. The two components are integrated in a joint model for deriving the evaluation cost. Extensive results on the three real-world datasets have shown the effectiveness and robustness of the proposed model.

2 RELATED WORK

Our work is related to the following three research directions.

Traffic Speed Prediction. With the development of traffic sensors and GPS-enabled devices, the task of traffic spped prediction has received much attention from the research community [8, 11, 12, 24, 26], which aims to predict the average vehicle speed of roads at a certain period of time. In the literature, various deep learning methods have been developed for traffic speed prediction, including recurrent neural network (RNN) [12, 26], convolution and graph convolution network (GCN) [11, 21, 24] and deep belief network (DBN) [8]. Traffic speed prediction model is an important component of our work, and the novelty lies on capturing long term temporal pattern and complex spatial influence by dilated causal convolution and direction-sensitive graph attention network.

Fastest Route Recommendation. With the availability of traffic condition information and corresponding road network, many efforts have been devoted to the task of fastest path recommendation [4, 6, 9, 15, 27–29]. This take aims to find the fastest path between the source and destination locations. In the literature, various search algorithms have been developed for route recommendation, including time-dependent search algorithm [27–29] and improved A^* algorithm [6, 9]. Early studies focus on modeling the travel time distribution of every road segment based on historical information with shallow models [6, 28, 29]. More recently, some researchers predict travel time of every road based on real time traffic information and find fastest path based on prediction [15, 27].

Machine Learning for Heuristic Search. These studies in this direction aim to automatically improve or optimize the search algorithms with machine learning methods. Early works include the use of machine learning in creating effective, likely-admissible or improved heuristics [5, 10, 17]. More recently, deep learning has significantly pushed forward the research of this line. The main idea is to leverage the powerful modeling capacity of neural networks for improving the tasks that require complicated solving strategies, including the Go game [19] and Atari games [13]. Our work is highly inspired by these pioneering works, but have a quite different focus on the studied task, *i.e.,* fastest route recommendation. Our task itself involves specific research challenges that make the reuse of previous works impossible.

3 PROBLEM FORMULATION

In our task, we assume road network information is available as input, which is the foundation of traffic communication for users.

Definition 1. ***Road Network.*** *A road network is a directed graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, where $\mathcal{L}$ is a vertex set of locations and $\mathcal{E} \subset \mathcal{L} \times \mathcal{L}$ is an edge set of road segments. A vertex $l_i \in \mathcal{L}$* (i.e., *a location) represents a road junction or a road end. An edge $e = \langle l_i, l_j \rangle \in \mathcal{E}$ represents a directed road segment from vertex l_i to vertex l_j.*

Definition 2. ***Time-Varying Road Speed.*** *Given a road segment e, its traffic speed $v_e(t)$ is modeled as time-varying variable, measuring the flow speed of traffic at time t.*

In mathematics, $v_e(t)$ can be viewed as a real-value function of road speed *w.r.t.* time t, in which t can be either continuous or

discrete. Here, for simplicity, we rewrite this notation as $v_{e,t}$ by assuming t is discrete.

Definition 3. ***Route**. A route (*a.k.a., *a path) p is an ordered sequence of road segments connecting the source location l_s with the destination location l_d with m intermediate locations,* i.e., $p : l_s \xrightarrow{e_0} l_1 \xrightarrow{e_1} \dots \xrightarrow{e_{m-1}} l_m \xrightarrow{e_{m-1}} l_d$, *where each pair of consecutive locations $\langle l_i, l_{i+1}\rangle$ corresponds to a road segment $e_i = \langle l_i, l_{i+1}\rangle$ in the road network.*

With the above prelimenaries, we now define the studied task.

Definition 4. ***Fastest Route Recommendation (FRR)**. Given a road network and corresponding historical traffic speed data $\mathcal{D}$, for a query $q : \langle l_s, l_d, t_{start}\rangle$, we would like to infer the fastest route p^* (*i.e., *the path with the least travel time) from l_s to l_d with the departure time t_{start} given the time-varying traffic condition.*

Different from traditional studies on fastest path recommendation [4, 6, 9, 15, 27–29], we do not assume a static traffic condition (*i.e.,* the road speeds of all the road segments are fixed or known beforehand). We aim to develop a more practical solution to the FRR task.

4 FASTEST ROUTE RECOMMENDATION WITH HEURISTIC A^* SEARCH

As shown in [4, 6, 9, 27–29], the task of FRR can be framed as a graph-based search problem. In this setting, we view the road network as a graph, and study how to find possible route(s) that start from the source node and end at the destination node.

Reviewing A^* Algorithm. In the literature [7], A^* search algorithm is widely used in pathfinding and graph traversal due to its performance and accuracy. Starting from a source node of a graph, it aims to find a path to the given destination node resulting in the smallest *cost*. It maintains a tree of paths originating at the source node and extending those paths one edge at a time until its termination criterion is satisfied. At each extension, A^* evaluates a candidate node n based on a *cost function* $f(n)$

$$f(n) = g(n) + h(n), \tag{1}$$

where $g(n)$ is the cost of the path from the source to n (we call it *observable cost* since the path is observable), and $h(n)$ is an estimate of the cost required to extend the future path to the goal (we call it *estimated cost* since the actual optimal path is unknown). The key part of A^* is the setting of the heuristic function $h(\cdot)$, which has an important impact on the final performance.

Setting $g(\cdot)$ and $h(\cdot)$. In our task, it is relatively straightforward to set the $g(\cdot)$ function by summing over the time costs of all the observable road segments. Assume a partial route has been generated, *i.e.,* $p : l_s \to l_1 \cdots \to l_i$, we can compute the observable cost of a candidate l_{i+1} for extension as

$$\begin{aligned} g(l_s \to l_{i+1}) &= \sum_{k=1}^{i} \text{Time}(l_k \to l_{k+1}|t_k, q), \\ &= C_{l_s \to l_i} + \text{Time}(l_i \to l_{i+1}|t_i, q), \end{aligned} \tag{2}$$

where $C_{l_s \to l_i}$ is a constant value since the trajectory has been observed. To set the above formula, we need to estimate the time cost from l_i to l_{i+1} at departure time t_i, *i.e.,* $\text{Time}(l_i \to l_{i+1}|t_i, q)$. Furthermore, to set $h(\cdot)$, we need to estimate the future cost from l_{i+1} to the destination l_d, *i.e.,* $h(l_i \to l_d)$. In practice, many heuristics have been adopted to set both $g(\cdot)$ and $h(\cdot)$, including the shortest spatial distance [7, 14] and the historical likelihood [22].

Potential Issues. A key component of A^* algorithm is the future cost function $h(\cdot)$. Unlike other traffic-related tasks, the FRR task is rather time dependent since the traffic condition varies over time. It is likely that a fast route at office time becomes extraordinarily crowded during rush hours. A possible solution is to incorporate time-related heuristics to improve the search algorithms. However, it will be very difficult to characterize the complex temporal factors using simple heuristics. Without effectively modeling dynamic traffic condition, it would be difficult to accurately estimate the future cost function $h(\cdot)$. Besides, the setting of $g(\cdot)$ also relies on the estimation of $\text{Time}(l_i \to l_{i+1}|t_i, q)$, which also requires to model dynamic traffic condition on a road segment. Another issue is that traditional A^* algorithm is generally difficult to be extended. It is not clear how to effectively integrate modules for modeling time-varying traffic condition.

With these considerations, for the PRR task, we seek a more powerful approach to setting $g(\cdot)$ and $h(\cdot)$ functions, *i.e.,* using deep learning. We aim to provide more effective solutions to fastest route recommendation by modeling complex time-varying traffic condition.

5 OUR MODEL

In the section, we present the *Neuralized A-Star based Fastest route recommendation* model, denoted by *NASF*.

5.1 Model Overview

Our model is developed based on the general A^* algorithm framework. For node evaluation, we decompose the entire cost function $f(\cdot)$ into two parts, namely *observable cost* and *estimated cost*, which correspond to the cost functions $g(\cdot)$ and $h(\cdot)$. Traditionally, both $g(\cdot)$ and $h(\cdot)$ are heuristically computed or set. While, our idea is to automatically set the two functions with neural networks instead of using heuristics. Specially, we develop a core component that is able to predict time-varying road speed for each road segment in the road network at varying time. With the speed predictor, we can set the $g(\cdot)$ function in Eq. 2 by adding one-step forward time cost with previous time cost. To set $h(\cdot)$ function, we first generate top-K shortest paths in spatial distance, and develop a bi-directional GRU network for estimating the future cost. Once the two networks are learned, we can compute the cost of a candidate location for path extension. We present the overall architecture for the proposed model in Fig. 1.

5.2 Predicting Time-Varying Traffic Speed for Road Segments

To find the fastest route, it is essential to effectively estimate the traffic speed at varying time for each road segment.

5.2.1 Formalization. Here, we aim to predict the traffic speed through a road segment $e = \langle l_i, l_j\rangle$ at departure time t. As defined in Section 3, the traffic speed of a road segment can be formed

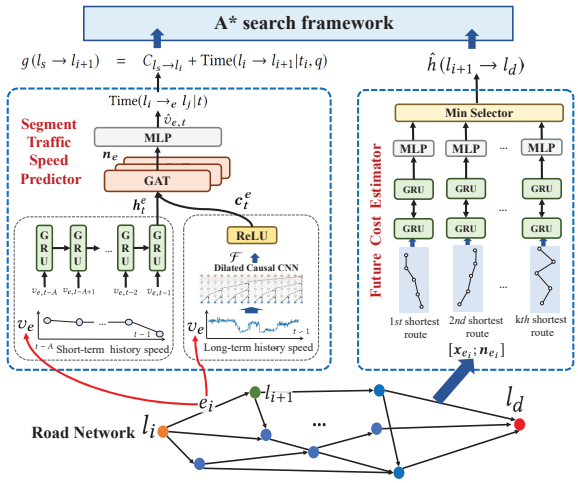


Figure 1: The overall architecture of the our model. $g(\cdot)$ learns the cost from the source to a candidate location, called *observable cost*; $h(\cdot)$ predicts the estimated cost from a candidate location to the destination, called *estimated cost.*

as time series $\{v_{e,t}\}_{t'=1}^{m}$ at discrete time steps. To estimate $v_{e,t}$, we assume the historical speed data (before time t) of all the road segments is available. It is challenging to predict time-varying traffic speed since the dynamics of a traffic system are rather complex. Many factors can potentially influence the final prediction performance, which cannot be effectively addressed by using simple statistical methods [9, 27–29]. Here, we identify three key elements to build our predictive model, namely short-term trend, long-term temporal patterns and spatial influence. Next, we discuss the speed estimation for an individual road segment. The estimation process can be repeated on all the road segments in parallel.

5.2.2 Modeling Short-term Trend with Recurrent Neural Networks. Given current time step t, we consider a backtracking period of t^S time steps in a short period, *i.e.,* from $t - t^S$ to $t - 1$. To model the short-term trend, we employ the GRU network to encode a subsequence of recent speed values (the data from the latest hour in our work). At time t, we update the hidden state of a road segment e via the GRU network as

$$\boldsymbol{h}_t^e = \text{GRU}(v_{e,t}, \boldsymbol{h}_{t-1}^e), \tag{3}$$

where $\boldsymbol{h}_t^e \in \mathbb{R}^{K_R}$ is the hidden vector produced by the GRU network and $v_{e,t}$ is the speed of road segment e at time step t. The vector $\boldsymbol{h}_t^e$ encodes the short-term trend of speed varying for road segment e. Subsequently, we will use the hidden vector $\boldsymbol{h}_t^e$ as part of the state representation of a road segment at time step t.

5.2.3 Modeling Long-term Temporal Patterns with Dilated Causal Convolutional Neural Networks. In addition to recent trend, it is also important to consider long-term temporal patterns. For example, the road speed may show a significant drop at rush hours, and the traffic flows more smoothly on weekdays. To identify such patterns, we need to develop a model that is able to capture temporal locality from a long-term data history. Here, we propose to use a convolution based neural network for learning such patterns. To ensure no leakage from the future into the past, we first apply causal convolution [1] to model historical traffic speed data. A disadvantage of simple causal convolution is that it requires a very deep architecture or very large filters to capture long-term information. Hence, we further apply the dilated convolution operation $\mathcal{F}$ [25] on the time series of $\{v_{e,1}, \cdots, v_{e,n}\}$ of a road segment e:

$$\mathcal{F}(\{v_{e,1}, \cdots, v_{e,n}\}) = \sum_{i=0}^{k-1} \text{filter}(i) \cdot v_{e,n-d\cdot i}, \tag{4}$$

where d is the dilation factor, k is the filter size, and $n - d \cdot i$ accounts for the direction of the past. According to [25], dilation is equivalent to introducing a fixed step between every two adjacent filter taps. When $d = 1$, a dilated convolution reduces to a regular convolution. Using larger dilation enables an output at the top level to represent a wider range of inputs, thus effectively expands the receptive field of traditional convolutional networks. Finally, we utilize a ReLU unit to generate the long-term representation from the CNN component for road segment e

$$\boldsymbol{c}_t^e = \text{ReLU}(\mathcal{F}(\{v_{e,1}, \cdots, v_{e,n}\}). \tag{5}$$

After we have learned the short-term representation $\boldsymbol{h}_t^e$ (Eq. 3) and long-term representation $\boldsymbol{c}_t^e$ (Eq. 4) , we concatenate the two representations as the representation of a road segment. Such a representation encodes necessary information that reflects the traffic characteristics of this road segment at time t, namely $[\boldsymbol{h}_t^e, \boldsymbol{c}_t^e]$.

5.2.4 Modeling the Spatial Influence with Direction-Sensitive Graph Attention Network. Intuitively, the road segments connected in road networks should be highly correlated in terms of traffic speed. For example, if the downstream road segment is in congestion, the traffic speed of current road segment would slow down. To capture the spatial influence, we utilize the recently proposed Graph ATtention network (GAT) [20] to characterize the influence of locations through the road network. Here, note that we treat *road segments* as graph nodes, and the junction between two road segments as the edge. The major reason is that we need to model the correlation of traffic speed among different road segments. Formally, the update of GAT can be given as

$$\boldsymbol{N}^{(z+1)} = \text{GAT}\left(\boldsymbol{N}^{(z)}|t\right), \tag{6}$$

where $\boldsymbol{N}^{(z)} \in \mathbb{R}^{K_G \times |\mathcal{E}|}$ denotes the matrix consisting of node representations at the z-th iteration, and the e_k-th column $\boldsymbol{n}_{e_k} \in \mathbb{R}^{K_G}$ corresponds to the representation of node e_k, *i.e.,* road segment $e_k \in \mathcal{E}$. Here, our GAT model aims to learn the dynamic representation of a road segment at time t. Note that z does not indicate the time step but the iteration number for a specific time t. For initialization, we set

$$\boldsymbol{n}_{e_k}^{(0)} = [\boldsymbol{h}_t^{e_k}, \boldsymbol{c}_t^{e_k}], \tag{7}$$

where we concatenate the learned representations at time t in Eq. 3 and Eq. 4. A key point in GAT is the setting of the attention weights between two nodes. In the road network, the edge direction is very important to consider, since the spatial influence between road segments is usually asymmetric. Hence, we set two kinds of

attention weights between two road segments l_j and $l_{j'}$ as

$$\alpha_{e_{j'}\to e_j} = \frac{\exp\left(\boldsymbol{w}_2^\top \cdot \left(\boldsymbol{W}_3\boldsymbol{n}_{e_j} + \boldsymbol{W}_4\boldsymbol{n}_{e_{j'}}\right)\right)}{\sum_{k\in\mathcal{I}_{e_j}} \exp\left(\boldsymbol{w}_2^\top \cdot \left(\boldsymbol{W}_3\boldsymbol{n}_{e_j} + \boldsymbol{W}_4\boldsymbol{n}_{e_k}\right)\right)}, \quad (8)$$

$$\beta_{e_j\to e_{j'}} = \frac{\exp\left(\boldsymbol{w}_2^\top \cdot \left(\boldsymbol{W}_3\boldsymbol{n}_{e_j} + \boldsymbol{W}_4\boldsymbol{n}_{e_{j'}}\right)\right)}{\sum_{k\in\mathcal{O}_{e_j}} \exp\left(\boldsymbol{w}_2^\top \cdot \left(\boldsymbol{W}_3\boldsymbol{n}_{e_j} + \boldsymbol{W}_4\boldsymbol{n}_{e_k}\right)\right)}, \quad (9)$$

where $\mathcal{I}_{e_j}$ and $\mathcal{O}_{e_j}$ are the upstream and downstream road segments of e_j, $\boldsymbol{W}_{(\cdot)}$ and $\boldsymbol{w}_2$ are learnable parameters, and $\alpha_{e_{j'}\to e_j}$ and $\beta_{e_j\to e_{j'}}$ are the attention weights from upstream and downstream nodes respectively. The major difference between the attention weights α and β lies in the normalization factor. We call such a model as Direction-Sensitive Graph ATtention network (DS-GAT). In trajectory datasets, it is easy to obtain or infer the direction of a road segment according to the traffic flow. If a road segment is bi-directional, we would equally add two uni-directional edges. For each uni-directional edge, we use $\alpha_{e_{j'}\to e_j}$ or $\beta_{e_j\to e_{j'}}$ according to the edge direction. To capture the complex spatial influence, we use the multi-head attention for enhancing the representations. We combine the results of A attention heads as

$$\boldsymbol{n}_{e_i}^{(z+1)} = \Big\|_{a=1}^{A} \text{relu}\left(\sum_{e_j\in\mathcal{I}_{e_i}} \alpha_{e_j\to e_i}^{(a)} \boldsymbol{W}^{(a)} \boldsymbol{n}_{e_j}^{(z)} + \sum_{e_j\in\mathcal{O}_{e_i}} \beta_{e_i\to e_j}^{(a)} \boldsymbol{W}^{(a)} \boldsymbol{n}_{e_j}^{(z)}\right), \quad (10)$$

where $\alpha_{i,j}^{(a)}$ are the normalized attention scores computed by the a-th attention head, "$\|$" denotes the concatenation operation and $\boldsymbol{W}^{(a)}$ is transformation weight matrix for the input. After $\boldsymbol{n}_{e_i}$ has been learned, we utilize a MLP-based predictor function to estimate the traffic speed of road segment e at time t

$$\hat{v}_{e,t} = \text{MLP}(\boldsymbol{n}_e). \quad (11)$$

As shown in Eq. 7, the learned node representation actually encodes the temporal information from the corresponding time. Note that we have omitted the time index t from $\boldsymbol{n}_e$ for simplicity. To train the model, we can compute the mean squared error loss

$$\mathcal{L}_1 = \sum_{\langle e,t\rangle\in\mathcal{D}} (v_{e,t} - \hat{v}_{e,t})^2, \quad (12)$$

where $\hat{v}_{e,t}$ is the predicted road speed via Eq. 11.

5.2.5 Setting the $g(\cdot)$ Function. Given a road segment, once the traffic speed has been predicted, we can divide the road length by the predicted traffic speed as the travel time through the road segment as follows:

$$\text{Time}(l_i \to_e l_j | t) = \frac{\text{length}(e)}{\hat{v}_{e,t}}. \quad (13)$$

In this way, given current location l_i, we can derive the value of $g(\cdot)$ in Eq. 14 by adding one-step forward time cost for the candidate location l_{i+1}:

$$g(l_s \to l_{i+1}) = C_{l_s\to l_i} + \text{Time}(l_i \to l_{i+1} | t_i, q). \quad (14)$$

5.3 Estimating the Fastest Arrival Time based on Top-K Shortest Routes

Compared with $g(\cdot)$, it is more difficult to set $h(\cdot)$, since it needs to estimate the future travel time (or the arrival time) from current location. Our idea is although the fastest path is not among top-K shortest routes, the traffic information of top K shortest routes provides important evidence for estimating the fastest arrival time.

5.3.1 Generation of Top-K Shortest Paths. Given a candidate location, we can efficiently generate the top-K shortest routes from it to the destination. Following method proposed in [3], we can use a time of $O(O(K \cdot |\mathcal{L}| \cdot (|\mathcal{E}| + |\mathcal{L}| \log |\mathcal{L}|)))$ to fulfill this step, where $|\mathcal{L}|$ and $|\mathcal{E}|$ are the number of locations and road segments in a road network, respectively. Here, we consider a route as a sequence of consecutive road segments. Let $e_1 \to e_2 \cdots \to e_n$ denote the generated path consisting of n road segments. In this way, we can better utilize the learned traffic state information through a sequence of road segments.

5.3.2 Estimation of the Fastest Arrival Time. Here, our task is to estimate the fastest arrival time from a candidate location at a departure time. We predict such a time based the K shortest routes from it to the destination. After generating K shortest routes, we build an arrival time predictor on top of bi-directional GRU (BiGRU) [18]. First, we construct the representation of each road segment e_i by concatenating two vectors $[\boldsymbol{x}_{e_i}; \boldsymbol{n}_{e_i}]$, where $\boldsymbol{x}_{e_i}$ encodes the attribute information of a road segment (including length, width and category) with a lookup layer, and $\boldsymbol{n}_{e_i}$ is the learned node representation using the GAT in Eq. 10. We employ BiGRU to capture both forward and backward temporal dependencies in a route. At each time step, we take as input the vector $[\boldsymbol{x}_{e_i}; \boldsymbol{n}_{e_i}]$, and update the corresponding state representation. Via the BiGRU, we can obtain the state representations of each road segment from a route in both forward and backward directions. We sum the two representations as the final representation of a road segment, *i.e.,* $\boldsymbol{s}_{e_j} = \overrightarrow{\boldsymbol{s}_{e_j}} + \overleftarrow{\boldsymbol{s}_{e_j}}$. We concatenate the representations of the first segment and the last segment as the representation of the k-th path, denoted by $\boldsymbol{z}^k \in \mathbb{R}^{Ks}$. We pass the path representation into a MLP component. Since there are K paths, we choose the minimum one of the K values:

$$\hat{h}(l_{i+1} \to l_d) = \text{Minimum}_{k=1}^{K}\left(\text{MLP}(\boldsymbol{z}^{(k)})\right). \quad (15)$$

Based on this equation, we further define the loss for the $h(\cdot)$ function:

$$\mathcal{L}_2 = \sum_{l_i,t} (h_{l_i,t} - \hat{h}_{l_i,t})^2. \quad (16)$$

where $\hat{h}_{l_i,t}$ is the predicted value in Eq. 15, $h_{l_i,t}$ is the actual travel time t denotes the departure time from l_i.

5.3.3 Admissibility Analysis. In heuristic search [7], a heuristic function is said to be *admissible* if it never overestimates the cost of reaching the goal, *i.e.,* the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path. If an admissible heuristic is adapted in an algorithm, then this algorithm would eventually find an optimal solution to the goal. In mathematics, it is difficult to directly verify the admissibility of our learned heuristics. Here, we present a simplified admissibility analysis under some reasonable assumptions. Formally, the probability

that our model has found the optimal path can be defined as:

$$p^* = \prod_{l_i=l_s}^{l_d} \Pr_u(\hat{h}(l_i \rightarrow l_d)), \quad (17)$$

$$= \prod_{l_i=l_s}^{l_d} \left(1 - \prod_{j=0}^{K} \Pr_o\left(\hat{h}(l_i^j \rightarrow l_d^j)\right)\right), \quad (18)$$

where $\Pr_u(\hat{h}(l_i \rightarrow l_d))$ and $\Pr_o(\hat{h}(l_i \rightarrow l_d))$ are the probabilities that underestimates or overestimates the cost of a path $l_i \rightarrow l_d$, respectively. We further assume $\Pr_o(\hat{h}(l_i \rightarrow l_d)) = p_o$ for all the locations in all the routes. In this case, we derive an estimate of K:

$$K \approx \log_{p_o}\left(1 - \sqrt[L]{p^*}\right), \quad (19)$$

where L is the length of the optimal path from l_s to l_d. This equation gives a rough estimation about the required number of shortest paths. Consider a running example for admissibility analysis. If we would like to be guaranteed by a large probability for p^*, say 0.9, with $p_o = 0.5$, $K \approx 6$ for an optimal path of length 10. In other words, we can achieve a probability of 0.9 to find the optimal route with six shortest paths. With suitable K, we can highly guarantee the the probability that finds the optimal path.

5.4 The Fastest Route Recommendation Algorithm

When the model parameters have been learned, we can apply the proposed model to generate fastest routes in response to users' queries. We follow the standard search procedure of A^* algorithms and perform the repeated selection of nodes with minimum cost to expand.

Algorithm Procedure. In implementation, we use priority queue as the fundamental data structure. Specially, we maintain two sets, called *closed set* (C) and *open set* (O). At each step, a node l^* with the minimum cost is excluded from the open set for expansion, and added into the closed set. Given a candidate location, we utilize Eq. 14 to compute the value for $g(\cdot)$ function, and utilize Eq. 15 to compute the value for $h(\cdot)$ function. Finally, the two cost values are summed as the final evaluation cost of a candidate location. We update the observable cost ($G[l']$) for each neighbor l' of l^* based on the RNN component. The entire cost ($F[l']$) is also updated by obtaining new estimated cost using the value network. The algorithm continues until a goal node has a lower f value than any node in the queue.

Model Analysis. We focus on a key problem for the PRR task, *i.e.,* the estimation of travel time. For estimating travel time, we identify a fundamental function that is required to be implemented, *i.e.,* the prediction of road speed. As shown in Fig. 1, we first design a speed predictor that is able to characterize short-term trend, long-term temporal patterns and spatial influence, and then develop a module to estimate the travel time between two locations in a road network conditioned on the speed predictor. Based on the two components, $g(\cdot)$ and $h(\cdot)$ are set to compute the observable cost and estimate the future cost respectively. In this way, the proposed model naturally combine the merits of traditional heuristic search algorithms and deep learning. Our model empowers traditional heuristic search algorithms with the capacities of modeling complex data patterns. For fastest route recommendation, online time complexity is important to consider. Compared with traditional A^*, the additional cost to evaluate a node includes the cost to identify the top-K shortest paths and the cost to make the computation based on neural networks. Although we can use accelerated shortest path finding algorithm, it still time-consuming. Our idea is that although the overall road network is highly connected, the reachable locations from a destination location are highly limited within a small step number (*e.g.,* 3 or 5 steps). We can offline compute all the top K shortest paths between any two reachable locations. Besides, we can also consider applying Monte Carlo sampling methods to generate K high-quality paths, which are approximately shortest paths. Once the model parameters have been learned, the computation cost of neural networks can be highly paralleled, which will be limited to a small time cost.

Algorithm 1 The search algorithm for our model.

```
procedure FastestRouteSearch(l_s, l_d, b)
    Initialize C, O, F, G
    while O is not empty do
        Obtain location l* ← O.pop()    ▷ location with the lowest F[]
        if l* = l_d then
            return the derived route from l_s to l_d
        end if
        O.remove(l*)
        C.add(l*)
        for neighbor l' ∈ L_{l*} do
            if l' ∈ C then
                continue
            end if
            G' ← G[l_c] + g(l_s → l')          ▷ Computed by Eq. (14)
            if l' ∉ O then
                O.add(l')
            else if G' ≥ G[l'] then
                continue
            end if
            G[l'] ← G'
            F[l'] ← G[l'] + h(l' → l_d)        ▷ Computed by Eq. (15)
        end for
    end while
end procedure
```

Parameter Learning. To learn the model parameters, we first need to train the model parameters with the $\mathcal{L}_1$ loss in order to learn the speed predictor. Then, we learn the model parameters via optimizing the $\mathcal{L}_2$ loss. The parameters in GRU, CNN, GAT and MLP are initialized by a truncated normal distribution with zero mean and 0.01 variance, and the biases are initialized as zeros. We use the Adaptive Moment Estimation (Adam) optimizer to train with a learning rate of 0.001. The batch size is set as 100 for the pre-training of $g(\cdot)$, and is set as 50 for the joint training of $g(\cdot)$ and $h(\cdot)$. The epoch number for the the pre-training is set as 50 and for the joint training is set as 20 until convergence. The number of shortest paths k is set as 6.

Table 1: Statistics of the three datasets after preprocessing.

Statistics	Beijing taxi	Q traffic	PEMSD7
Duration	1 month	2 months	1 month
Δinterval	1 minute	15 minute	5 minute
#records	569,678,400	11,139,840	22,101,120
#road segments	13,187	1,934	2,558
#locations	8,592	1,285	1,749
#querys	48,000	48,000	48,000

6 EXPERIMENTS

In this section, we first set up the experiments, and then present the performance comparison and analysis.

6.1 Experimental Setup

6.1.1 Construction of the Datasets. To measure the performance of the proposed model, we adopted three real-world traffic speed datasets in our experiment. Details of the three datasets are listed in Table 1.

The first traffic speed dataset was generated by GPS trajectories of 50,000 taxis in Beijing. We name this dataset as *BT-Traffic* (Bejing Taxi Traffic Speed). The taxi trajectories used in BT-Traffic were collected during April 1 to April 30, 2015. We use the *openstreetmap*[1] to acquire the road network of Beijing and match the trajectories on the road network using the open source tool *Fast Map Matching* [2]. The traffic speed of a road segment is set as the average speed of taxies that move past the segment during a sampling period. For the BT-Traffic dataset, the sampling period is set as every minute.

The second dataset is *Q-traffic*, which is a public traffic speed dataset released by the Baidu Map [12]. The Q-traffic dataset contains traffic speed of road segments in Beijing, China during April 1, 2017 to May 31, 2017. The dataset is sampled by every 15 minutes.

The third dataset is *PeMSD7*, which was collected by the Caltrans Performance Measurement System (PeMS) through over 39,000 sensors that are deployed across the major metropolitan areas of California state highway system [3]. The data of 2558 stations in the District 7 of California are used in our experiment. The time range is from May 1 to May 31, 2012. It is sampled by every 30 minutes.

6.1.2 Evaluation Metrics. For the FRR task, we adopt two *Fast Rate* metrics to evaluate the performance of our model:

$$FR1 = \frac{\#(\text{A's real travel time} < \text{the shortest travel time})}{\#queries},$$

$$FR2 = \frac{\text{A's real travel time} - \text{the shortest travel time}}{\text{best travel time}}.$$

Here, the shortest travel time is acquired using the algorithm proposed in [4], which uses the A* algorithm to find the fastest path with future traffic speed data available.

6.1.3 Task Setting. For each dataset, we keep the data in the first 70% days as a training set, and the data in the next 10% days as a validation set. The loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ in Eq. (12) and Eq. (16) are trained and optimized with the training and validation sets. The performance of FRR algorithms with optimized $h(\cdot)$ and $g(\cdot)$ functions was tested using the data in the rest 20% days. Over the test dataset, we generate three types of queries *w.r.t.* distances between source and destination, namely *short queries*, *medium queries*, and *long queries*. We set short queries as $< 5\ km$, medium queries as $5 - 10\ km$, long queries as $> 10\ km$ for the Beijing taxi dataset, set short as $4 - 5\ km$, medium as $5 - 6\ km$, and long as $> 6\ km$ for the Q-traffic dataset, and set short as $< 20\ km$, medium as $20 - 50\ km$, and long as $> 50\ km$ for the PeMSD7 dataset. For queries in each type, we vary their departure time as each hour of one day. The average performance of each type of queries are reported.

6.1.4 Methods to Compare. We consider the following methods as baselines to compare:

•*STATIC:* The baseline finds shortest path based on traffic speeds of all roads at the departure time of a query. It equals to find the shortest path on a static travel time graph at a departure time.

•*T-drive* [29]: This baseline mines smart driving behaviors from the historical GPS trajectories of a large number of taxi drivers, and provides users with the practically fastest route to a given destination at a given departure time.

•*IAFP* [9]: This baseline adopts a novel extension of the A* algorithm to find a time-dependent fastest route with historical traffic speed data of road networks.

•*ARIMA*: In this baseline, we use the AutoRegressive Integrated Moving Average model [2] to predict traffic speed $v_{e,t}$ in the function $g(\cdot)$, and set $h(\cdot) = 0$ for the algorithm 1.

•*SVR*: In this baseline, we use the Support Vector Regression model [23] to predict traffic speed $v_{e,t}$ in the function $g(\cdot)$, and set $h(\cdot) = 0$ for the algorithm 1.

•*STGCN*: In this baseline, we use a novel deep learning framework, Spatio-Temporal Graph Convolutional Networks (STGCN) [24], to predict traffic speed $v_{e,t}$ in the function $g(\cdot)$, and set $h(\cdot) = 0$ for the algorithm 1.

Among these baselines, STATIC only uses the current (departure time) traffic speed to plan a route. T-drive and IAFP are historical traffic information based methods, which use experiential traffic speed to plan a route. ARIMA, SVR and GRU are baselines that have proactive ability to predict traffic speed and plan a route.

6.2 Results and Analysis

We present the results of all the comparison methods in Table 2. First, from the table we can that see the NASF model is consistently better than all the baselines in all cases. This result verified the performance advantage of the proposed fastest route recommendation algorithm.

Second, for the lack of real time traffic information, historical data based methods, *i.e.,* IAFP and T-drive, perform not very well. Compared with IAFP, the performance of T-drive is better. The reason might be T-drive has an ability to estimate the distribution of travel time for each road segment while IAFP can only estimate traffic speeds of segments as discrete levels. Intuitively, modeling the travel time distribution should be more effective compared with only constructing coarse-grained speed levels.

Third, the STATIC method sometimes is better than IAFP and T-drive, especially for the short and medium distance queries. A

[1] https://www.openstreetmap.org
[2] https://www.github.com/cyang-kth/fmm
[3] http://pems.dot.ca.gov/

Table 2: Performance comparison using two metrics on three datasets. With paired t-test, the improvement of the our model over all the baselines is significant at the level of 0.01.

Datasets	Metric	BT-Traffic			PeMSD7			Q-traffic		
	Length	short	medium	long	short	medium	long	short	medium	long
FR1	T-Drive	0.499	0.513	0.548	0.316	0.319	0.324	0.352	0.366	0.371
	IAFP	0.512	0.535	0.582	0.322	0.326	0.330	0.377	0.383	0.391
	STATIC	0.388	0.533	0.612	0.236	0.247	0.260	0.311	0.374	0.433
	ARIMA	0.364	0.437	0.494	0.228	0.239	0.253	0.295	0.353	0.421
	SVR	0.318	0.402	0.458	0.213	0.225	0.241	0.283	0.331	0.403
	STGCN	0.264	0.310	0.395	0.202	0.217	0.229	0.258	0.301	0.387
	NASF	**0.251**	**0.293**	**0.358**	**0.189**	**0.201**	**0.215**	**0.216**	**0.257**	**0.305**
FR2	T-Drive	0.112	0.147	0.214	0.094	0.098	0.102	0.119	0.122	0.127
	IAFP	0.132	0.167	0.234	0.108	0.111	0.115	0.128	0.131	0.136
	STATIC	0.082	0.141	0.231	0.053	0.062	0.073	0.072	0.104	0.133
	ARIMA	0.076	0.133	0.220	0.048	0.057	0.063	0.067	0.088	0.118
	SVR	0.071	0.125	0.199	0.044	0.050	0.057	0.065	0.082	0.115
	STGCN	0.061	0.118	0.211	0.041	0.051	0.056	0.062	0.078	0.108
	NASF	**0.040**	**0.103**	**0.166**	**0.030**	**0.041**	**0.047**	**0.037**	**0.047**	**0.081**

possible reason is that traffic conditions might not change very significantly for short distance trips, so in this condition, the current traffic speed is more close to actual traffic speed compared with historical traffic speeds. Therefore, STATIC using current traffic speed to plan routes may have better performance than IAFP and T-drive. However, the experiment results also show that, for long distance trips, where traffic conditions might change very significantly, using historical traffic speed is very necessary.

Lastly, the prediction based baselines, *i.e.,* ARIMA, SVR and STGCN, have good performance compared with STATIC and historical data based baselines. This result indicates the proactive ability is very important for the FRR problem. Comparing the three prediction based baselines, we can see the performance is $STGCN > SVR > ARIMA$. As shown in Section 6.3.2, the traffic speed prediction performance of the three baselines is in the same order, indicating that a better traffic speed prediction performance is very helpful for efficient fastest route planning.

By summarizing these results, we can see the models with traffic speed prediction abilities are competitive to solve the FRR task, especially when a good traffic speed prediction model is adopted. Our model combines both heuristic search and neural networks to solve the FRR problem, and the proposed traffic speed prediction model can fully exploit the information in long/short-term historical data and road network structure information, so it performs the best among the comparison methods.

6.3 Detailed Analysis on our Model

In this section, we perform a series of detailed analysis on our model for further verifying its effectiveness. Due to space limit, we only report the results on the BT-traffic dataset. The rest results show the similar findings, and are omitted here.

6.3.1 Effect of Model Modules in FRR.

We first examine the effect of different modules to FRR performance. Here, we consider four cropped versions of our model: (1) *DC*, which is our model without graph attention network and GRU for short-term trend modeling, *i.e.,* only the Dilated Causal CNN module. (2) *GRU*, which is our model without graph attention network and dilated convolution modules, *i.e.,* only the GRU module. (3) *¬GAT*, which is our model without the graph attention network module. (4) *¬DS*, which is our model without direction sensitivity module. The whole model is denoted as NASF. The *FR*1 performance of the four variants are listed in Table 3. From the table we can see that the performance rank follows $DC < GRU < \neg GAT < \neg DS < NASF$. The performance of *GRU* is better than *DC*, which indicates the short-term trend information is more important than the long-term trend information. The model performances are improved by graph attention network and direction sensitivity modules, which verified the effectiveness of the two modules.

Table 3: FR1 Comparison among Model Modules

Query	Short	Medium	Long
DC	0.322	0.365	0.484
GRU	0.295	0.339	0.423
¬GAT	0.268	0.314	0.401
¬DS	0.257	0.301	0.387
NASF	**0.251**	**0.293**	**0.358**

Table 4: Effectiness of Traffic Speed Prediction Modules

Interval	5 min	15 min	30 min	60 min
ARIMA	7.31	9.14	10.45	12.13
SVR	6.29	8.28	9.14	10.57
¬GAT	4.20	5.36	5.93	6.98
STGCN	4.13	5.21	5.87	6.69
¬DS	4.06	5.18	5.71	6.57
NASF	**4.02**	**4.97**	**5.59**	**6.37**

6.3.2 Analysis of Model Traffic Prediction Performance. As we discussed in Section 6.2, the performance of traffic speed prediction is especially important for our model. Here, we examine the prediction performance of the traffic speed prediction modules, *i.e.,* the GRU, DC, GAT, and DS modules. In the experiment, we prepare

three cropped versions of our model: (1) ¬*GAT* using our traffic speed prediction model without the GAT and DS modules. (2) ¬*DS* using our traffic speed prediction model without the DS module, *i.e.,* considering road network as undirected graph. (3) *NASF* using all modules. Moreover, three traffic prediction models are used as baselines: (1) *ARIMA* using ARIMA model to predict future speed. (2) *SVR* using support vector regression to predict future speed. (3) *STGCN* using spatio-temporal graph convolution neural network.

Following [24], we use the *MAP* (Mean Average Precision) of all road segments for all time periods as metric to evaluate speed prediction performance of our model. The speed unit is $km/hour$. In Table 4, it can be observed that the performance rank is as follows: *ARIMA* < *SVR* < ¬*GAT* < *STGCN* < ¬*DS* < *NASF*. Although the ¬*GAT* does not use any road network structure and neighbor segment traffic information, its performance is very close to the *STGCN* where the road network information is involved. Both ¬*DS* and *NASF* exploit the traffic speed information of neighbor segments, and then they achieved the improved performance compared with *STGCN*. Moreover, the performance of *NASF* is better than ¬*DS*, indicating that the direction information in road network is useful for traffic speed prediction.

Table 5: Admissibility of NASF.

Query	1km	3km	5km	10km	15km
AEE	-0.531	-0.931	-1.531	-3.016	-5.51
OE	6.72%	6.33%	5.82%	5.57%	5.31%
OP	89.79%	77.68%	71.82%	66.54%	62.39%

6.3.3 Analysis of Admissibility. Admissibility is a key factor in the design of heuristic. If heuristic is admissible, it will make sure the A* algorithm to find the optimal path. As detailed in [7], if the heuristic function is admissible, it never overestimates the actual cost to get to the goal. Here, we use three metrics to evaluate admissibility of our model. (1) *AEE*: Average Estimation Error for all future travel time estimations. (2) *OE*: the percentage of estimations that overestimate the fastest arrival time. (3) *OP*: percentage of queries that can find the optimal path. In the experiments, the query distance is varied from $1km$ to $15km$. The experiment results are listed in Table 5. It can be observed that AEEs of our model for all conditions are negative, which indicates the heuristic of our model is "on average" admissible. The OE for all estimations are less than 7%, which indicates that for most of cases our model is admissible. From the OP performance we can see that our model can find the optimal path most of the time.

Table 6: Search Space of Heuristics Networks (# Expanded Roads)

Query	1km	3km	5km	10km	15km
ED	31.8	94.3	254.2	1066.5	1555.4
¬DC	18.9	53.7	161.3	619.8	855.1
¬GRU	17.7	49.3	151.1	605.3	825.6
NASF	13.5	41.2	125.5	519.5	716.0

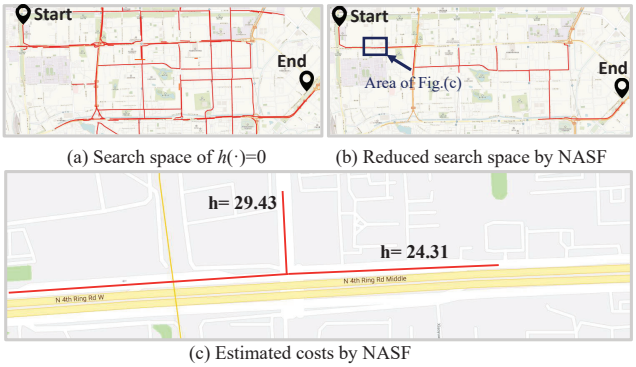
Figure 2: Visualization of traffic speed information in road networks. A darker color indicates a lower traffic speed.

6.3.4 Analysis of Search Space. Another important metric to evaluate the heuristic function of a search algorithm is the size of search space. Reduced search space can result in a faster inference speed. In this part, we prepare four model variants for the search space comparisons, including (1) *ED* using Euclid distance as heuristics, (2) ¬*DC* using our heuristic network without the dilated causal convolution module, (3) ¬*GRU* using our heuristic network without the GRU module, (4) *NASF* using our whole model. The average expanded road number of these methods for varied query distances are listed in Table 6. It can be observed that the performance rank is as follows: *ED* < ¬*DC* < ¬*GRU* < *NASF*. We can see that the simplest spatial distance baseline *ED* gives the worst performance, which indicates simple heuristics does not work well in our task. ¬*DC* < ¬*GRU* < *NASF* indicates that both long-term and short-term traffic trends can help our model to reduce search space.

6.4 Qualitative Analysis

Previously, we have shown the effectiveness of our model in the FRR task. In this part, we qualitatively analyze why our NASF model is able to yield a good performance by some show cases.

In our model, the traffic prediction module is a core module for FRR. Figure 2 is a show case to demonstrate how the traffic speed prediction helps our model to generate a faster route recommendation than other methods. In the show case, a FRR algorithm tries to give a route recommendation from the start point to the end point. As shown in Fig. 2(a), the route recommended by the T-drive algorithm is along the gray lines on left lower part of the map, which is the shortest path from start point to the end point through expressways. However, our NASF model gives a different recommendation, which is along the blue lines on upper right part of the map. Although the physical distance of our model's recommendation is longer than T-drive's recommendation, the experiment result shows the actual travel time of our model's recommendation ($13min$) is shorter than T-drive's recommendation ($22min$). Fig. 2(b) is a traffic speed heat map after 15 min. As shown in the figure, traffic congestion happened on the road of T-drive's recommendation. The NASF model predicted the congestion in advance and then



(a) Search space of $h(\cdot)$=0 (b) Reduced search space by NASF

(c) Estimated costs by NASF

Figure 3: Visualization of the search procedure with the estimated costs by the NASF model.

recommended user to avoid the congestion. That is why our model achieved an improved fastest rout recommendation performance.

Next, we continue to study how the $h(\cdot)$ function helps the search procedure in NASF. Figure 3 presents a show case. Here, given the start and end points, the model tries to find the fastest route. The red lines on the Fig. 3(a) denote the segments that are searched by NASF with $h(\cdot) = 0$, and on Fig. 3(b) denote searched by the whole model. By comparing the two figures, it can be seen that the $h(\cdot)$ of our model is able to effectively reduce the search space. When zooming into a subsequence of this route, we further compare the estimated $h(\cdot)$ for two candidate road segments (red line) in Fig. 3(c). It's obvious that turn left will make user away from the destination. The figure shows that the $h(\cdot)$ function gives a longer estimated arrival time, *i.e.,* 29.43 > 24.31, to avoid this pointless search.

7 CONCLUSIONS

In this paper, we took the initiative to adopt neural networks to automatically learn the cost functions in A^* for the FRR task. To improve the FRR task, we focused on the estimation of travel time by modeling complex traffic information with neural networks. For this purpose, we developed an estimation module for the travel time conditioned on a well-designed speed predictor for road segments. We constructed extensive experiments for verifying the effectiveness and robustness of the proposed model.

Since road network information is not always available, as future work, we will consider extending our model to solve the FRR task without road networks. Now, we focus on the FRR task. We will study whether our solution can be generalized to other search tasks.

ACKNOWLEDGMENTS

The work was supported by National Key Research and Development Program of China under Grant No. 2017YFC0820405 and National Natural Science Foundation of China under Grant No. 61572059, 61872369, 71531001. Zhao's work was partially supported by the Research Funds of Renmin University of China under Grant No. 18XNLG22 and the Fundamental Research Funds for the Central Universities. Wu's work was partially supported by the Science and Technology Project of Beijing under Grant No. Z.181100003518001. Wang's work was partially supported by Open Research Program of Shenzhen Key Laboratory of Spatial Smart Sensing and Services.

REFERENCES

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. Convolutional Sequence Modeling Revisited. (2018).
[2] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. 1976. Time series analysis: Forecasting and control. Rev. ed. *Journal of Time* 31, 4 (1976), 238–242.
[3] Lijun Chang, Xuemin Lin, Lu Qin, Jeffrey Xu Yu, and Jian Pei. 2015. Efficiently Computing Top-K Shortest Path Join. (2015), 133–144.
[4] Bolin Ding, Jeffrey Xu Yu, and Lu Qin. 2008. Finding time-dependent shortest paths over large graphs. In *EDBT*. ACM, 205–216.
[5] Marco Ernandes and Marco Gori. 2004. Likely-admissible and sub-symbolic heuristics. In *ECAI*. Citeseer, 613–617.
[6] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. 2007. Adaptive fastest path computation on a road network: a traffic mining approach. *very large data bases* (2007), 794–805.
[7] Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.
[8] Yuhan Jia, Jianping Wu, Moshe Ben-Akiva, Ravi Seshadri, and Yiman Du. 2017. Rainfall-integrated traffic speed prediction using deep learning method. *IEEE TITS* 11, 9 (2017), 531–536.
[9] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. 2006. Finding fastest paths on a road network with speed patterns. In *ICDE*. IEEE, 10–10.
[10] Levi Lelis, Roni Stern, and Shahab Jabbari Arfaee. 2011. Predicting solution cost with conditional probabilities. In *ASSC*.
[11] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).
[12] Binbing Liao, Jingqing Zhang, Chao Wu, Douglas McIlwraith, Tong Chen, Shengwen Yang, Yike Guo, and Fei Wu. 2017. Deep Sequence Learning with Auxiliary Information for Traffic Prediction. In *SIGKDD*.
[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
[14] Karl Nachtigall. 1995. Time depending shortest-path problems with applications to railway networks. *EJOR* 83, 1 (1995), 154–166.
[15] Xiaoguang Niu, Ying Zhu, Qingqing Cao, Xining Zhang, Wei Xie, and Kun Zheng. 2015. An Online-Traffic-Prediction Based Route Finding Mechanism for Smart City. *International Journal of Distributed Sensor Networks* 11, 8 (2015), 970256.
[16] Stefano Pallottino and Maria Grazia Scutella. 1998. Shortest path algorithms in transportation models: classical and innovative aspects. In *Equilibrium and advanced transportation modelling*. Springer, 245–281.
[17] Mehdi Samadi, Ariel Felner, and Jonathan Schaeffer. 2008. Learning from Multiple Heuristics.. In *AAAI*. 357–362.
[18] Mike Schuster and Kuldip K Paliwal. 1997. *Bidirectional recurrent neural networks*. IEEE Press. 2673–2681 pages.
[19] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.
[20] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* 1, 2 (2018).
[21] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method. (2016), 499–508.
[22] Ling Yin Wei, Yu Zheng, and Wen Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *SIGKDD*. 195–203.
[23] Chun Hsin Wu, Jan Ming Ho, and D. T. Lee. 2004. Travel-Time Prediction With Support Vector Regression. *IEEE TITS* 5, 4 (2004), 276–281.
[24] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *ijcai* (2018).
[25] Fisher Yu and Vladlen Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. *international conference on learning representations* (2016).
[26] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. 2017. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 777–785.
[27] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. (2011), 316–324.
[28] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2013. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE TKDE* 25, 1 (2013), 220–232.
[29] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*. ACM, 99–108.