2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Learning Urban Navigation via Value Iteration Network

Shu Yang, Jinglin Li*, Jie Wang, Zhihan Liu and Fangchun Yang

Abstract— **Choosing an appropriate route is a critical problem in urban navigation. Being familiar with roads topology and other vehicles' routes, experienced drivers could usually find a near optimal route. However, the nowadays navigation applications hardly catch the domain knowledge and drivers' interaction in this scenario. In fact, they only recommend several routes and leave the most difficult decision to driver. Hence the route is often congested by many vehicles whose drivers make a similar choose. To intelligently make right decision on navigation and improve traffic efficiency for each vehicle, we propose a neural network structure which learns to plan coarse-grained route in complex urban areas.**

Focused on learning to navigate, this paper first formalizes urban map and vehicle route model. The city map is segmented into grids and each vehicle's route is mapped to grids. Based on grid-world model, we solicit both global traffic status and driving actions from large-scale taxicab GPS data. The learn-to-plan problem is therefore to find a policy function from a global status representation to an experienced driver's action under that status. Traditional neural network is difficult to learn to this plan-involved function, so we leverage and modify value iteration network (VIN), which explicitly takes long-term plan into consideration. Finally we evaluate the performance of proposed network on real map and trajectory data in Beijing, China. The results show that VIN can achieve human driver performance in most cases, with high success rate and less commuting time.

I. Introduction

The rise of computer science and technology fuels the research of intelligent transportation system. Rapid progress in deep learning not only help vehicles to recognize object [1], [2], but also enable them to make better control according to complicated environment [3]–[6]. Furthermore, connected vehicles are able to communicate and collaborate in the city [7]–[9], thus efficiency of whole transportation system could be optimized by sharing global spatiotemporal information or by coordinating individuals' routes.

A promising service in intelligent transportation system is route planning or navigating for smart vehicles. Being networked and intellectualized, smart vehicles are able to coordinate in different levels to reduce commuting time [10]. The nowadays navigation apps, such as Google, Baidu and Amap, usually collect information throughout city, but fail to coordinate vehicles. The reason is intuitive, these apps only recommend several routes to driver, leaving driver to decide which route to take. However, quite a few drivers choose a "looked like" shortest road, only to find the route is congested by many vehicles whose drivers make a similar decision. This phenomenon is referred to "congestion game" [11], [12] in social and economic science.

To tackle this problem, we train an experienced "brain" to navigate in coarse-grained map. Such brain takes as input the global traffic status and the destination, then produces a next driving action towards the destination. This brain uses special neural network structure and is trained by plenty of experienced driving data. In our envision, each vehicle should equipped a brain to make experienced and rational route decision. Concentrated on making neural network learn to plan, this paper has following three contributions.

Problem formalization of learning to plan. The map is segmented into grids and each vehicle's route is mapped to the map grids. We therefore build a grid-world which could reflect urban traffic flow if appropriate grid granularity is selected. We get a global "traverse time map (TTM)" by merging vehicle GPS and speed data. Additionally, we solicit discrete action sequence from vehicle trajectories, aligning each action to a "traverse time map" by time stamps. Overall, we build a formalized data repository including millions of driving tuple=$(TTM, action)$. The data contains abundant information which helps neural network learn to plan. The learning problem therefore transforms into finding the policy function mapping from TTM set to action set.

Structure design of neural network. Because of the internal complexity of planning, the policy from TTM to action could not be well learned by traditional neural network, such as Fully Connected Network (FCN) or Convolutional Neural Network (CNN). Instead, we use Value Iteration Network (VIN), a fully differentiable neural network with a "planning module" embedded within. VIN can learn to plan, and are suitable for predicting actions that involve planning-based reasoning, such as optimal route planning. To provide appropriate information for VIN to learn, the input TTM and VIN are co-designed to capture urban transportation features.

Network training and performance evaluation. To validate planning ability of VIN, real road networks and vehicle trajectories in Beijing are used. VIN is fed with millions of driving tuples and trained end-to-end. We show that by learning an explicit planning computation, VIN can navigate well and largely achieve human driver performance with high success rate and less commuting time.

The rest of this paper is organized as follows: Section.II describes background and basic framework of navigation learning. Section.III describes how we formalize problem for learning, and presents details of VIN. We demonstrate

This work is supported by the National Science and Technology Major Project of China under Grant No.2016ZX03001025-003 and Special fund for Beijing Common Construction Project.

*Corresponding author: Jinglin Li. All authors are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. Email: {assureys, jlli, jayw, zhihan, fcyang}@bupt.edu.cn

Project is available on github.com/JackWangCS/VIN-Urban-Navigation

the effectiveness of VIN in Section.IV. The paper ends in Section.V, with conclusion and discussion on navigation learning.

II. Preliminaries

In this section, we first provide background on navigation, value iteration and planning. Then we depict our framework designed for navigation learning.

A. Background

Navigation: Over decades, GPS-assisted equipment has come to pervade many aspects of intelligent transportation system. When driving to a place wanted to go, it can accurately locate your position, rapidly calculate an optimal route, and then give correct guidance through voice and image. The optimal route computation is the key in the above process. Most navigation approaches can obtain the exact route plan in math measure. However, the exact results from Dijkstra's algorithm are sometimes no accord with that from the experienced drivers [13]. In fact, traditional navigation app only recommends several routes and leaves the most difficult decision to driver. However, quite a few drivers usually choose a "looked like" shortest road, only to find the route is congested by many vehicles whose drivers make a similar choose. Being familiar with roads topology and other drivers' routes, experienced drivers could usually find a near optimal route, which is not exactly the mathematical shortest route but a time-saving one. There has been effort to develop anticipatory navigation system which can act as a driver's assistant and helps in the evaluation of alternate routes [14]. Such a system can strongly influence driver's choice and make vehicles participate global coordination.

Value iteration: A standard model for sequential decision making and planning is the Markov Decision Process (MDP) [15]. An MDP M consists of states $s \in \mathcal{S}$, action $a \in \mathcal{A}$, a reward function $R(s,a)$, and a transition matrix $P(s'|s,a)$ that encodes the probability of the next state given the current state and action. A policy $\pi(a|s)$ describes an action distribution for each states. The goal in an MDP is to find a policy that obtains high rewards in the long-term. Formally, the value $V^\pi(s)$ of a state under policy π is the expected discounted sum of rewards when starting from that state and $\mathbb{E}^\pi[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s_0=s]$, where $\gamma \in (0,1)$ is a discount factor, and $\mathbb{E}^\pi$ denotes an expectation over trajectories of states and actions $(s_0,a_0,s_1,a_1,...)$, in which actions are selected according to π, and states evolve according to the transitional matrix $P(s'|s,a)$. The optimal value function $V^*(s) \doteq max_\pi V^\pi(s)$ is the maximal long-term reward possible from a state. A policy π^* is said to be optimal if $V^{\pi^*} = V^*(s)$ for $\forall s$. A popular algorithm for calculating V^* and π^* is the Value Iteration (VI):

$$V_{n+1}(s) = max_a Q_n(s,a) \forall s, \text{where}$$
$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s'). \quad (1)$$

The value function V_n in VI converges to V^* as $n \to \infty$, from which an optimal policy maybe derived as $\pi^*(s) = argmax_a Q_\infty(s,a)$. It is interesting to adapt MDP and VI into our grid-world navigation scenario and exploit learning the optimal driving policy. We build on the work from Tamar *et al.* [16] who study how explicit planning can be incorporated in neural network structure, but do not consider the case in large-scale city.

DeepNav: The research most directly relevant to our work is DeepNav [18], a convolutional neural network based algorithm for navigating large cities using locally visible street-view images. Some structures are shared across environments - for example, most cities will have restaurants in business district, or gas stations are usually found near highway exits. Knowledge of such structure can be used to navigate even driver has no map of the environment. Inspired by this, DeepNav let agent learn to reach its destination quickly by making the correct navigation decisions at intersections without giving explicit map. The agent neither has a map of the city, nor does it know the destination or itself. What agent can use is some "common sense" learned from data. The authors collect a large-scale dataset of street-view images organized in a graph, and supervise the agent by grouptruth. Similar to DeepNav, our work also aims to navigate in big cities. However, our work still builds a global status representation for navigation while DeepNav does not.

B. Framework

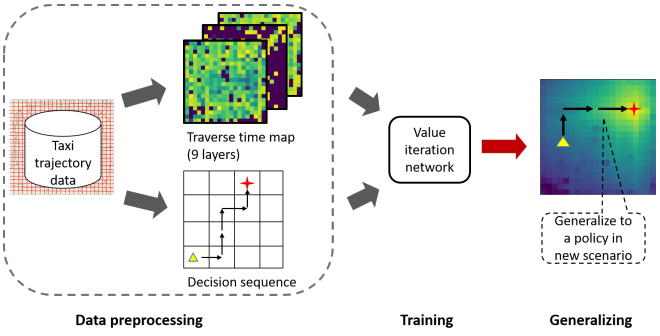

Fig. 1. A framework of navigation learning. It consists of three processes: data preprocessing, training and generalizing.

In order to learn navigation from taxi trajectory data, our framework consists of three processes. Fig. 1 is a description of framework of navigation learning. **Data preprocessing**: The map of city is segmented into $M \times N$ grids. Trajectory data from experienced drivers are fed into two information pipelines. One pipeline merges speed data to produce traverse time map, and the other discretizes each trajectory into driving actions (decision sequence) in a grid-world view. **Training**: Traverse time maps, aligning with actions by time stamps, are fed into VIN. VIN are taught to make an *action* given a *traverse time map*. Such data are extracted from experienced taxi drivers when they were carrying passengers, VIN therefore learns something useful from them. We co-design the structure of traverse time maps and actions so that VIN could be trained end to end. **Generalizing**: After

training, VIN is allowed to give a route when faced with a new scenario (traverse time map).

III. Value Iteration Network For Urban Navigation

A. Building MDP

Definition1 - Grid Map: Our navigation is coarse-grained therefore the urban map is modeled as grid-based graph $G = (V, E)$, where the grid is denoted as vertex V and nearby grids in 8 directions are reachable through virtual edge E. The set of grid is denoted by $V = \{v_{0,0}, v_{0,1}, ..., v_{x,y-1}, v_{x,y}, ...,\}$, where subscripts x and y depict grid's location in two-dimensional coordinates. An edge weight e_{x_1,y_1,x_2,y_2} represents the consumed time driving from v_{x_1,y_1} to an adjacent v_{x_2,y_2}.

Definition2 - Traverse Time Map: Considering that one vehicle in grid $v_{x,y}$ could transit to 8 adjacent grids or simply stay at original grid, the edge set $\mathcal{E}$ should consist of 9 layers, $\mathcal{E} = \{E^0, ..., E^l, ..., E^8\}$. For any $e^l_{x,y}$ in E^l, it represents the traverse time from $v_{x,y}$ to $v_{x+dx(l),y+dy(l)}$ in the l^{th} direction. The traverse time map (TTM) $\mathcal{E}$ can be obtained by merging historical GPS-based trajectories. Our layered model is an approximation, we hope the transition among grids could somehow reflect roads topology and traverse time map could reflect traffic flow speed. Fortunately, this approximation works fine if appropriate grid granularity is selected.

Definition3 - Driving Action Sequence: Each occupied-taxi trajectory is mapped into grid map and discretized into driving action sequence set $\{a_0, a_1, ..., a_i, ...\}$, where $a_i \in \mathcal{A} = \{l_{0\sim8}\}$ represents a choice among 9 actions.

After aligning pair of TTM and action according to time stamps, we have built an MDP which grasps the essence of driving planning: An MDP M consists of states $s = v$, action a, a reward function $R(s, a) = R(v, a) = e^a_v \in \mathcal{E}$, and a transition matrix $P(s'|s, a) = P(v'|v, a)$ that encodes the probability of the next grid given the current grid and action. While these rewards and transitions are not necessarily the true rewards and transitions in navigation, an optimal plan in M will still follow a driving action sequence that along the route and reaches the destination, similar to the choice of experienced drivers.

Once MDP of navigation has been specified, standard planning algorithm can be used to obtain the value function V^*. An additional property of V^* in navigation scenario is that the optimal decision $\pi^*(s)$ at a state s can only depend on a subset of the value of V^*, since

$$\pi^*(s) = argmax_a[R(s,a)+\gamma \sum_{s'} P(s'|s,a)V^*(s')]. \quad (2)$$

And the transition has a local connectivity structure, the states for which $P(s'|s, a) > 0$ is a small subset of S, *i.e.* only 8 adjacent grids or ego grid s' are reachable to s.

B. Learning structure co-design

We co-designed data structure and VIN, which incorporates a differentiable planning computation. The main observation is that each iteration of value iteration could be seen as passing previous value function V_n and reward function R through a convolution layer and max-pooling layer. Therefore by recurrently applying convolution and max-pooling K times, K iterations of value iteration are performed. Near optimal driving policy could be derived from a converged value function V_n if n is big enough.

From this idea, we proposed value iteration network module, as depicted in Fig. 2. The input to the VIN module is a multi-channel 'reward map' R of dimensions l,x,y, where l represents directional layer, and x, y denotes one location in grid-world. The rewards are fed into a convolution layer Q with $\|\mathcal{A}\| = 9$ channels.

Each channel in this layer corresponds to $Q(s, a)$ for a particular driving action a. This layer is then max-pooled along the 9-action channel to produce the next-iteration value function layer V. The next-iteration value function layer V is then stacked with multi-channel reward map, and fed back into the convolutional layer and max-pooling layer K times, to execute K iterations of value iteration.

Once multi-channel reward and VIN are co-designed, the training data can flow from traverse time map to driving action straightforward. This enables VIN to be trained end to end.

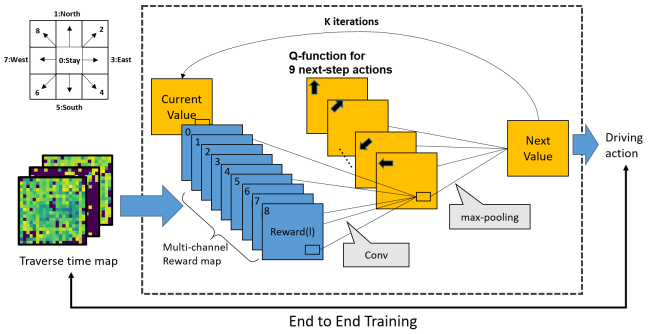

Fig. 2. Value iteration network module.

C. Training algorithm of VIN

The full algorithm for training VIN is presented in Algorithm 1. The initial neural network learns by batches. Because using historical data of arbitrary length as inputs can be difficult, our VIN instead works on a fixed length representation of historical tuples. We store the drivers' experiences at each time stamp, $tuple_t = (TTM_t, a_t)$, in a dataset $\mathcal{D} = \{tuple_1, ..., tuple_t, .., tuple_m\}$. The time stamp is coarse-grained and counted in every 10 minutes. The dataset $\mathcal{D}$ is shuffled and divided into many mini-batches. During the inner loop of the algorithm, we apply mini-batch gradient descent to update network parameters θ.

IV. Experiment

We illustrate the experiment from following three processes.

Algorithm 1: Algorithm for training VIN

Initialize Data set $\mathcal{D} = \{tuple_1, tuple_2, ..., tuple_m\}$
Initialize action-value function V with weights θ
for *epoch=1,N* **do**
    Initialize mini-batch number $num = m/batch\_size$
    for *i=0,num* **do**
        $j = i + batch\_size$
        $fd\_data = tuple_{i:j}$
        Perform a gradient descent step $\Delta\theta$ on $loss = Cross\_Entropy\{fd\_data.a, V(fd\_data.TTM, fd\_data.s; \theta)\}$ with respect to VIN parameters $\theta \leftarrow \theta + \Delta\theta$

A. Data preprocessing

Grid map: The grid map (V, E) is built from Beijing map within 4^{th} ring-highway. The detailed information of map is trivial for coarse-grained navigation, therefore only grid model is built in our experiment. Each grid has its position and traverse directions.

Taxi trajectory data: We extracted routes from real taxi trajectory data collected by MSRA [19]. The data package includes over 10000 taxicabs' trajectories in November 2013. For each day the data package contains a full-scale GPS during 24 hours. Since trajectory is too detailed for use, we simplified the GPS trajectory into grid-based route by trajectory-grid-matching, *i.e.* mapping GPS trajectory into sequence of grids, and tagged each grid with arriving/leaving time. Since our goal is to learn (near) optimal route in city, and "no passenger" taxicabs probably have no optimal route pattern, we only store the trajectory of taxicabs who are in "carrying passenger" state.

Traverse time map: For a certain grid $v_{x,y}$, a traverse direction $l_{1\sim8}$ and a fixed period $[t, t + \triangle T]$, we extract traverse time of all passing by taxicabs, average them to get one "reward grid" $e^{l,t}_{x,y} = -average\ time$ in TTM. Some grids' value $e^{l,t}_{x,y}$ are blank because of lacking trajectory data, their values are set a large negative reward indicating that grid $e_{x,y}$ in l^{th} direction is likely impassible.

Once an 8-layers TTM is calculated, the time stamp $[t, t + \triangle T]$ could be removed since driving action a only depends on global status representation $e^{l}_{x,y}$ regardless of time. In this way, TTM from different hours and days can be directly mixed for training. To get good performance in mini-batch gradient descent, TTM is normalized into $[-1, 0]$ by equation:

$$TTM_{norm} = \frac{TTM - TTM_{max}}{TTM_{max} - TTM_{min}}. \tag{3}$$

For l_0 layer, *i.e.* staying layer of TTM, we manually built the reward map. Since the destination (x_d, y_d) of each driving action is already known, $e^0_{x_d,y_d}$ is set $+1$, indicating a big reward for arriving destination. Other cells in e^0 are set -1.

Finally, $l_{1\sim8}$ and l_0 are concatenated into a 9-layers TTM, which contains both global status as well as destination information. As Fig. 3 presented, 9-layers TTM is calculated and fed to VIN, which later gives out a learned value map. By gradually moving from dark grid (far from destination) to bright yellow grid (destination), one can find routes that VIN has learned.

B. Training

The state space S was chosen to be a $m \times n$ grid-world. The reward R in this space can be represented by an $l \times m \times n$ traverse time map. The transition matrix P are defined as 3×3 convolution kernels in value iteration model, exploiting the fact that transitions in grid-based map are local.

Our training set consists of $Num_i = 500$ historical grid-world status representations, with $Num_j = 100$ destinations and $Num_t = 3$ driving actions; a total of $Num_i \times Num_j \times Num_t$ training instances. For each action in trajectory, we produce a $(9 \times M \times N)$-sized traverse time map. For 20×20 grid-world, the VIN has depth of $K = 30$, $epochs = 200$ and $batch\_size = 64$. For larger grid-world, such as 30×30, a bigger K should be set to make iteration converge.

C. Generalizing results

To have intuitive sight of VIN's navigating ability, we visualized trajectories for some typical cases in Fig. 4. The VIN starts from some grids and is required to reach the destination (grid in bright yellow). VIN generates a learned value map, which suggests a predicted route by iteratively choosing next optimal state.

The first figure presented ring-highway topology of Beijing. Fig. 4(b)-(n) show some success cases. We see that the agent is able to traverse large distances across multiple grids to get to the destination, keep off impassible blocks (parks, traffic jams). An interesting observation is that VIN's trajectory accords with Beijing's ring-highway. This illustrates that VIN works like an experienced driver who makes the most of ring-highway system. Fig. 4(o)-(r) show several typical failure cases, VIN fails to reach destination, probably due to wrong choices at first steps.

Tamar *et al.* have shown that VIN can generalize better than CNN or FCN in long-term planning problem [16]. Therefore, we only concentrated on several aspects of VIN and did not compared with other types of neural networks. In Table I, we demonstrate the performance by four indicators.

Accuracy: The top-1 accuracy is not high because of intrinsic uncertainty of navigation. Even experienced drivers in the same place could choose different routes to the same destination. In Beijing, the main roads are usually in *east-west* or *north-south* direction, hence there are usually more than one route lead to the same destination. This situation is similar to case in Fig.4(b), one route firstly goes south then goes east while another route goes east first and then goes south. In other words, the existing bayes error [17] of navigation makes it slightly difficult to get high top-1 accuracy. On the other hand, the high top-2 accuracy

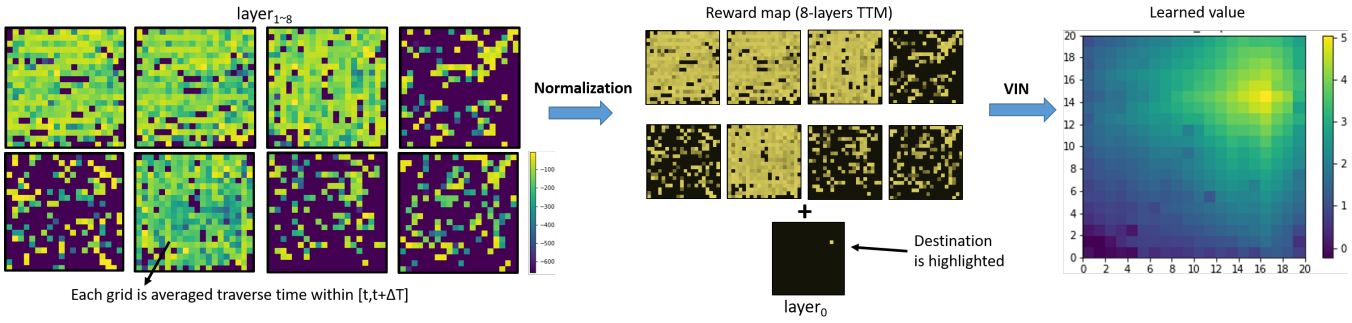

Fig. 3. Prepare TTM for VIN.

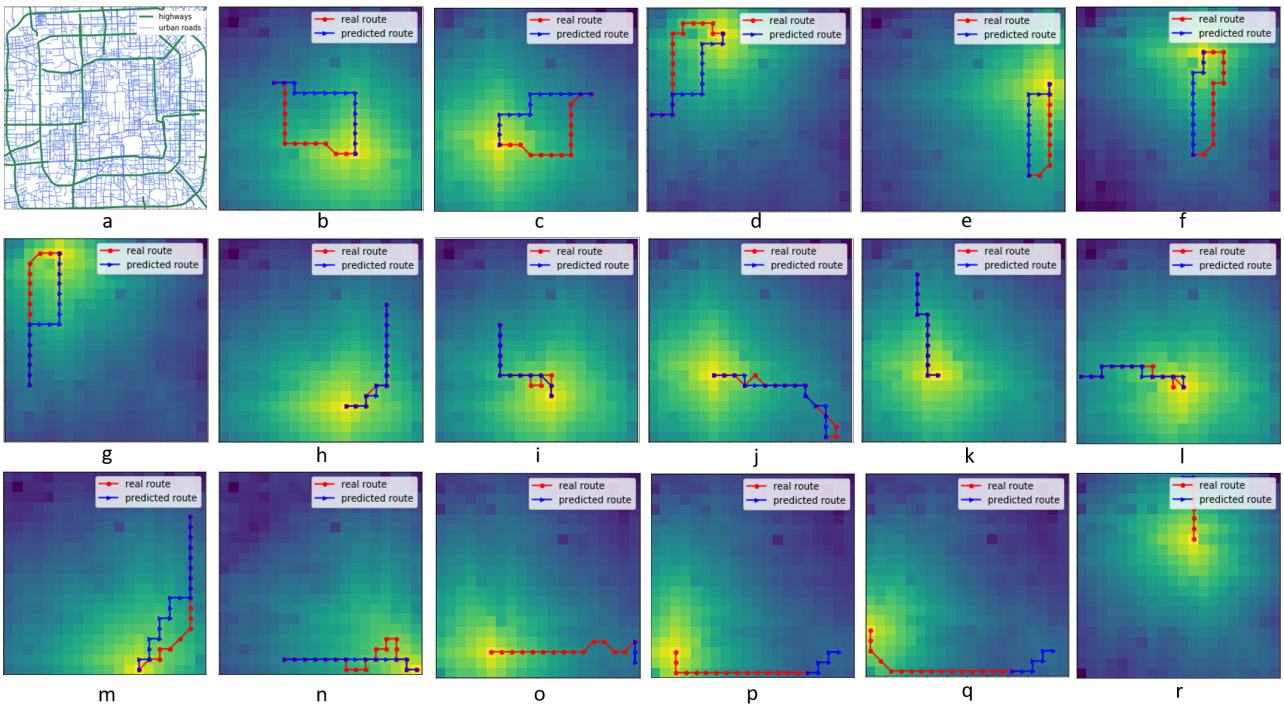

Fig. 4. Visualization cases. (a):Ring-highway topology of Beijing. (b)-(n):VIN has successfully reached destination with varies route choices. With a little GPS offset error, one could find that predicted trajectories accord with Beijing's ring-highway in (a). (o)-(r):VIN fails to navigate, probably due to wrong choices at first steps.

TABLE I
PERFORMANCE OF VIN

Domain	top-1 Accuracy	top-2 Accuracy	Success Rate	Saved Time Rate (only for success trails)
20×20	60.3%	79.5%	99.8%	9.3%
30×30	65.7%	83.8%	97.8%	18.4%
40×40	67.4%	86.1%	84.2%	24.0%

indicates that our VIN has successfully learned most driving pattern considering alternative choices.

Success rate: Given a global status TTM, a full trail from initial state is predicted by iteratively choosing the optimal next states. A trail is successful if it reaches destination via the predicted trajectory. Success Rate is the ratio between successful trail and all trail numbers. Our high success rate demonstrates that VIN is able to navigate in coarse-grained urban model.

Saved time rate: We randomly select real trajectories to build a sample set $\mathbb{T} = \{(O_1, D_1, time_1), (O_2, D_2, time_2), ..., (O_i, D_i, time_i), ...\}$, and use VIN to predict corresponding new trajectory set $\mathbb{T}' = \{(O_1, D_1, time'_1), (O_2, D_2, time'_2), ..., (O_i, D_i, time'_i), ...\}$. Note that we only consider success trajectories. We give "TTM invariant assumption" that changing one's

trajectory does not make difference on global TTM, so that TTM can be used to estimate total time-consuming of new trajectory. More specifically, applying VIN to a trajectory $T_i = (O_i, D_i, time_i)$ would produce a new trajectory $T_i' = (O_i, D_i, time_i')$, which saves time $\Delta time_i = time_i - time_i'$. Saved time rate (STR) is calculated from whole sample set $\mathbb{T}$:

$$STR = \frac{\sum_{T_i \in \mathbb{T}} \Delta time_i}{\sum_{T_i \in \mathbb{T}} time_i}. \quad (4)$$

Our result shows that a single vehicles could saved up to 24% commuting time, if using VIN to navigate in urban areas. One may argue that high penetration of vehicle equipped with VIN would change the internal structure of TTM where "TTM invariant assumption" no longer holds. This is an interesting topic to be investigated in the future.

Granularity. We found that most navigation patterns could be well approximated if appropriate grid granularity 30×30 is selected. Generally, model with finer granularity has higher predicting accuracy, but has lower success rate than coarser model. The reason is three-fold: (1)Finer model could grasp some traffic structure which coarser model ignores, so it achieves higher accuracy. (2)One trajectory in finer model consists of more steps than coarser model. Even finer model has higher top-2 accuracy, its success rate may still decrease since it needs more steps to predict. (3)Finer granularity would suffer more than coarser granularity model given the constant GPS noise. Overall, the opposite tendency between predicting accuracy and success rate suggests that we need to select appropriate grid granularity.

V. Conclusion and Discussion

This paper has discussed an end-to-end neural network to learn navigation in urban areas. It receives global status representation, *i.e.* traverse time map, and decides the next move in a simplified grid city. By feeding experienced drivers' data, our proposed scheme is able to achieve human driver performance in most cases, with high success rate and less commuting time. While our work represents progress towards navigation problems which have not been looked at from a machine-learning perspective, a lot more needs to be done for solving the problem of urban navigation.

A central limitation in our work is the assumption of grid-based traffic topology. Vehicles driving in the real world do not move grid by grid. Fortunately, we found that most navigation patterns could be well approximated if appropriate grid granularity is selected. Even though coarse-grained navigation is effective and helpful for drivers, the VIN can be further extended to a general discrete state spaces, such as normal road graph, in our future work.

Another related limitation is that traverse time map contains lots of noise. Traverse time of a region depends on dynamic of local traffic, which varies from time to time. The fluctuation of GPS data also deteriorates VIN's performance. However, as more and more smart vehicles participate urban sensing [20], along with localization technique improvements, traverse time map could be clearer and more accurate.

In this work, we do not consider traffic diversion explicitly since congestion avoiding is naturally rooted in experienced drivers' pattern. However, we envision that traffic diversion could be explicitly incorporated into VIN by substituting *max-pooling layer* with *stochastic-pooling layer* [21] in our architecture. Stochastic-pooling tries to add some uncertainty in pooling process thus being able to diverse traffic flow to several routes. To put our proposed method into real application, the "brain" should periodically provide the path with global status changing over time.

References

[1] Dai, Jifeng, et al. "R-fcn: Object detection via region-based fully convolutional networks." Advances in neural information processing systems. 2016.

[2] Wang, Panqu, et al. "Understanding convolution for semantic segmentation." arXiv preprint arXiv:1702.08502 (2017).

[3] Yu, Hao, et al. "Baidu driving dataset and end-to-end reactive control model." Intelligent Vehicles Symposium (IV), IEEE, 2017.

[4] Chen, Chenyi, et al. "Deepdriving: Learning affordance for direct perception in autonomous driving." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[5] Bojarski, Mariusz, et al. "End to end learning for self-driving cars." arXiv preprint arXiv:1604.07316 (2016).

[6] Zhu, Yuke, et al. "Target-driven visual navigation in indoor scenes using deep reinforcement learning." arXiv preprint arXiv:1609.05143 (2016).

[7] Han, Shuangshuang, et al. "Parallel vehicles based on the ACP theory: Safe trips via self-driving." Intelligent Vehicles Symposium (IV), 2017 IEEE. IEEE, 2017.

[8] Wang, Fei-Yue. "Agent-based control for networked traffic management systems." IEEE Intelligent Systems 20.5 (2005): 92-96.

[9] Bazzan, Ana LC. "Opportunities for multiagent systems and multi-agent reinforcement learning in traffic control." Autonomous Agents and Multi-Agent Systems 18.3 (2009): 342-375.

[10] Varaiya, Pravin. "Smart cars on smart roads: problems of control." IEEE Transactions on automatic control 38.2 (1993): 195-207.

[11] Kok, Adrianus Leendert, E. W. Hans, and J. M. J. Schutten. "Vehicle routing under time-dependent travel times: the impact of congestion avoidance." Computers & operations research 39.5 (2012): 910-918.

[12] Farokhi, Farhad, and Karl H. Johansson. "A study of truck platooning incentives using a congestion game." IEEE Transactions on Intelligent Transportation Systems 16.2 (2015): 581-595.

[13] Li, Qingquan, Zhe Zeng, and Bisheng Yang. "Hierarchical model of road network for route planning in vehicle navigation systems." IEEE Intelligent Transportation Systems Magazine 1.2 (2009): 20-24.

[14] Chen, Kan, and Steven E. Underwood. "Research on anticipatory route guidance." Vehicle Navigation and Information Systems Conference, 1991. Vol. 2. IEEE, 1991.

[15] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.

[16] Tamar, Aviv, et al. "Value iteration networks." Advances in Neural Information Processing Systems. 2016.

[17] Yang, Shuang Hong, and Bao-Gang Hu. "Discriminative feature selection by nonparametric bayes error minimization." IEEE Transactions on knowledge and data engineering 24.8 (2012): 1422-1434.

[18] Brahmbhatt, Samarth, and James Hays. "DeepNav: Learning to Navigate Large Cities." arXiv preprint arXiv:1701.09135 (2017).

[19] Yuan, Jing, et al. "T-drive: driving directions based on taxi trajectories." Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems. ACM, 2010.

[20] Zhu, Yanmin, Xuemei Liu, and Yin Wang. "Pervasive urban sensing with large-scale mobile probe vehicles." International Journal of Distributed Sensor Networks 2013 (2013).

[21] Zeiler, Matthew D., and Rob Fergus. "Stochastic pooling for regularization of deep convolutional neural networks." arXiv preprint arXiv:1301.3557 (2013).