Lane Change Scheduling for Autonomous Vehicle: A Prediction-and-Search Framework

Shuncheng Liu[1], Han Su[1,2*], Yan Zhao[3], Kai Zeng[4], Kai Zheng[1,2*]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
[2]Yangtze Delta Region Institute, University of Electronic Science and Technology of China, Quzhou, China
[3]Department of Computer Science, Aalborg University, Aalborg, Denmark
[4]Alibaba Group, Hangzhou, China
liushuncheng@std.uestc.edu.cn,hansu@uestc.edu.cn,yanz@cs.aau.dk
zengkai.zk@alibaba-inc.com,zhengkai@uestc.edu.cn

ABSTRACT

Automation in road vehicles is an emerging technology that has developed rapidly over the last decade. There have been many inter-disciplinary challenges posed on existing transportation infrastructure by autonomous vehicles (AV). In this paper, we conduct an algorithmic study on when and how an autonomous vehicle should change its lane, which is a fundamental problem in vehicle automation field and root cause of most 'phantom' traffic jams. We propose a prediction-and-search framework, called *Cheetah* (Change lane smart for autonomous vehicle), which aims to optimize the lane changing maneuvers of autonomous vehicle while minimizing its impact on surrounding vehicles. In the prediction phase, Cheetah learns the spatio-temporal dynamics from historical trajectories of surrounding vehicles with a deep model (GAS-LED) and predict their corresponding actions in the near future. A global attention mechanism and state sharing strategy are also incorporated to achieve higher accuracy and better convergence efficiency. Then in the search phase, Cheetah looks for optimal lane change maneuvers for the autonomous vehicle by taking into account a few factors such as speed, impact on other vehicles and safety issues. A tree-based adaptive beam search algorithm is designed to reduce the search space and improve accuracy. Extensive experiments on real and synthetic data evidence that the proposed framework excels state-of-the-art competitors with respect to both effectiveness and efficiency.

CCS CONCEPTS

• **Applied computing** → **Transportation**; • **Information systems** → *Spatial-temporal systems*; • **Computing methodologies** → *Planning and scheduling*.

*Corresponding authors: Kai Zheng and Han Su.

KDD '21, August 14–18, 2021, Virtual Event, Singapore.

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467072

KEYWORDS

Lane change maneuvers; Autonomous vehicle; Trajectory prediction

ACM Reference Format:
Shuncheng Liu[1], Han Su[1,2*], Yan Zhao[3], Kai Zeng[4], Kai Zheng[1,2*]. 2021. Lane Change Scheduling for Autonomous Vehicle: A Prediction-and-Search Framework. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467072

1 INTRODUCTION

With the rapid growth in urbanisation and ownership of cars, most major cities around the world suffer from high rates of traffic congestion, resulting in significant fuel waste and air pollution [28]. In everyday life, one may often encounter traffic jams without any obvious reason such as accidents, roadwork, or closed lanes, which are commonly known as 'phantom' traffic jams. It usually occurs when the traffic density is high, so even a small perturbation (e.g., forced lane change or hard braking) on traffic flow may generate 'butterfly effect' and cause serious traffic jams [32]. To prevent phantom traffic jams, it requires all vehicles on road to maintain a proper gap in-between and avoid improper lane changes or hard braking, which is very difficult, if not impossible, for human drivers.

With the rapid development of vehicle automation technology, this goal may be achieved in the future when a considerable portion of on-road vehicles are autonomous vehicles (AV). Existing studies have shown that with the help of Adaptive Cruise Control (ACC) for car-following control, smooth acceleration and braking, autonomous vehicles can maintain a constant gap to the preceding vehicle, thus mitigating the phantom traffic jam problem [11, 31]. However, as for the Lane Change Assistant (LCA) of autonomous vehicles, the existing methods mainly focus on the safety and comfortableness of lane-change maneuver [5, 32, 35], leaving the impact on other surrounding vehicles and eventually traffic conditions largely uninvestigated. For example, the manual of Tesla (Model S) warns drivers not to use Auto Lane Change on city streets or on roads with changing traffic conditions [29]. Apparently, without taking the driving behaviors of surrounding vehicles into account, it may cause more serious traffic jams or even accidents if autonomous vehicles decide when and how to change lane solely based on their own status. Therefore, at the core of our work is the assumption

that performing safe, smooth and efficient lane changes and reducing its impact on surrounding vehicles are both essential tasks for the LCA systems of autonomous vehicles.

To sum up, hard braking and improper lane changes are the two main reasons for most phantom traffic jams. In the field of autonomous driving, the first issue was studied more extensively and handled by ACC systems. However, the second issue has not been addressed well due to a few challenges: 1) Lack of proper framework; 2) Uncertainty of surrounding vehicles' driving behaviors; 3) Variety of vehicle maneuvers. In this work, we aim to address the above challenges and propose a new LCA system that fulfills SAE (Society of Automotive Engineers) level 4 or 5 automation [18].

For the simplicity of study, we consider a traffic scenario where there is one autonomous vehicle A and many conventional vehicles (with human drivers) $\mathbb{C}$ traveling on a multi-lane road. Our system can only control the behaviors of the autonomous vehicle: lane change, speed-up, speed-down, maintain speed. For other conventional vehicles, we can collect their real-time locations via Internet of Vehicles (IoV) or the autonomous vehicle's sensors. Our objective is to maximize the average speed of the autonomous vehicle without disturbing the other conventional vehicles as much as possible. To this end, we proposed a novel prediction-and-search framework, called Cheetah, which consists of two phases. In the prediction phase, a trajectory prediction model, namely GAS-LED (Global Attention and State sharing based LSTM Encoder-Decoder), is proposed to model the dependencies of spatial and temporal features from neighboring vehicle's historical trajectories and forecast their future behaviors. The hidden state sharing technique coordinated with global attention is designed to connect the encoder and the decoder, which not only achieves high accuracy but also reduces training time. Then during the search phase, Cheetah hunts for optimal lane change maneuvers efficiently with adaptive beam search algorithm based on the maneuver tree. Instead of setting the width parameter (e.g., beam width) heuristically, our search method can automatically select high-confidence nodes and prune low-confidence nodes in each layer of the tree.

To the best of our knowledge, this work provides the first data-driven solution to schedule the lane change maneuvers for autonomous vehicles by taking contextual vehicles into account. Our technical contributions can be summarized as follows:

- We develop a prediction-and-search framework that enables autonomous vehicle to perform lane change efficiently and smart, by minimizing its impact on other surrounding vehicles.
- We propose a trajectory prediction model (GAS-LED) that is equipped with a global attention mechanism and state sharing to improve the prediction accuracy and training efficiency.
- We propose a maneuver tree-based adaptive beam search algorithm to find the optimal maneuver sequence efficiently.
- We conduct extensive experiments to evaluate our framework on real and simulated data, verifying the effectiveness and efficiency on multiple metrics.

2 PROBLEM STATEMENT

In this section, we briefly introduce a set of preliminary concepts about Cheetah, based on which an overview of our problem and proposed framework are presented.

2.1 Problem Setting

The existing LCS systems mainly focus on the safety and comfortableness of driving, but ignore the impact of the lane change maneuvers of the autonomous vehicle on its surrounding conventional vehicles, which may cause traffic jams. In this work, we try to reduce this impact as much as possible. More formally speaking, **our objective is to maximize the average speed of the autonomous vehicle while minimizing its impact on surrounding conventional vehicles.**

In this study, we consider an interactive environment where there are one autonomous vehicle A and a set of conventional vehicles $\mathbb{C}$ driving on a straight multi-lane road. For the sake of simplicity, parking and turning are not considered for now. The autonomous vehicle can obtain the real-time location of surrounding vehicles through its sensors or IoV (Internet of Vehicles), and make decisions on lane change maneuver at each time instant within a target time duration $\mathbb{T}$ of interest. Without loss of generality, we assume the dimensions and performance of all vehicles are the same.

In the scenario mentioned above, we first define the notion of **lane**. A lane L is part of the road used to guide vehicles in the same direction. Usually, a road has multiple (at least two) lanes. Herein, all the lanes are numbered incrementally from the leftmost side to the rightmost side, e.g., $L_1, L_2, \cdots$ where L_1 indicates the leftmost lane. An advantage of using this type of lane-based coordination system is to allow us to focus on the lane change behavior itself without worrying about the lateral position of the vehicle. Next, we introduce two basic units for spatial and temporal dimensions respectively.

Trajectory Point. A trajectory point p indicates the location of a vehicle in a 2-dimensional space, wherein $p.L$ denotes the lateral lane number and $p.D_{lon}$ refers to the longitudinal distance of the vehicle traveled from the starting point of a road, which is a fixed point denoted as p_s i.e., $p_s.D_{lon} = 0$. $p^t_{C_i}$ or p^t_A is used to denote the trajectory point of the vehicle C_i or A at time instant t. The $d(p^t_{C_i}, p^t_{C_j})$ denotes the longitudinal distance between two trajectory points, which can be calculated as follows:

$$d(p^t_{C_i}, p^t_{C_j}) = |p^t_{C_i}.D_{lon} - p^t_{C_j}.D_{lon}| \tag{1}$$

where $p^t_{C_i}.D_{lon}$ is the longitudinal distance of vehicle C_i at time t, which equals to $d(p_s, p^t_{C_i})$.

Time Step. In order to model the problem more concisely, we treat the continuous time duration as a set of discrete time steps, i.e., $\mathbb{T} = \{1, 2, ..., t, t+1, ...\}$. In this study, we set the time interval between two consecutive time steps to 0.5 seconds, which is analyzed in Appendix.

Trajectory. A trajectory $\mathcal{T}$ consists of a sequence of vehicle's trajectory points, ordered by time step t, i.e., $\mathcal{T} = (p^1, p^2, \cdots, p^t)$. We use $\mathcal{T}_{C_i}$ and $\mathcal{T}_A$ to denote the trajectory of conventional vehicle C_i and autonomous vehicle A respectively.

Lane Change Maneuver. A maneuver is a series of (almost) simultaneous behaviors performed by a vehicle in order to accomplish a task (e.g., change lane). Inspired by recent studies in microscopic traffic field, we represent a lane change maneuver at time t as a pair of a lateral lane change behavior and a longitudinal motion behavior, i.e., $M_t = \langle B_{lc}, B_m \rangle$. B_{lc} can be one of the three behaviours: *change left* (Lt), *change right* (Rt), and *do not change*

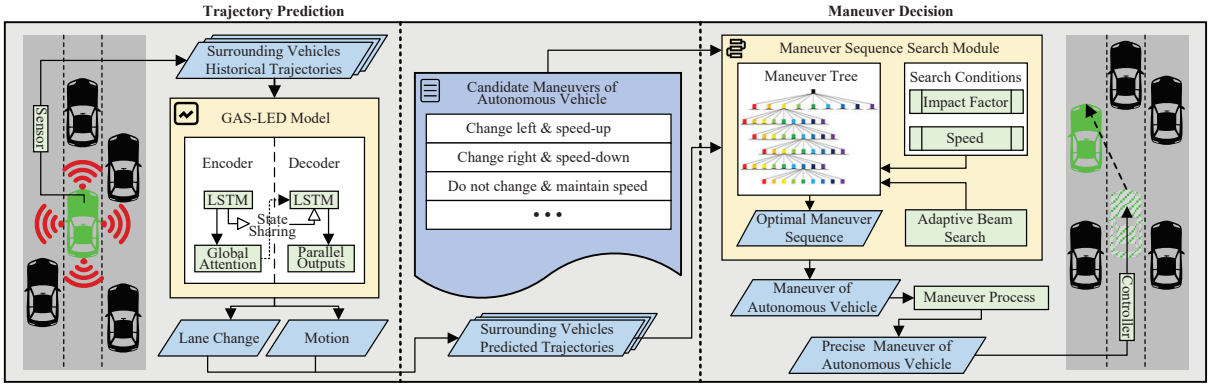

Figure 1: Cheetah overview

(Nt), i.e., $B_{lc} \in \{Lt, Rt, Nt\}$. B_m indicates the longitudinal distance travelled by the vehicle between current and the next time step, i.e., $B_m = d(p^t, p^{t+1})$.

Given a time duration of interest $\mathbb{T}$, the autonomous vehicle will perform a sequence of lane change maneuvers, denoted by $\mathcal{M} = \{M_1, M_2, ..., M_t\}$, with the objective to maximize its average speed and minimize the impact on other conventional vehicles. In this paper, we use maneuver to stand for lane change maneuver whenever the context is clear.

Restrictions. We pose some traffic restrictions on all the vehicles:
• Speed limit: all lanes are subject to two speed limits: V_{min} and V_{max}.
• Lane change restrictions: a vehicle can only change to adjacent lane at each time step. Besides, it needs to keep D_{lcs} distance at minimum with the preceding and tailing vehicles in the target lane.
• Safe following distance: all vehicles in the same lane should keep a safe following distance D_{ss}.

2.2 Framework Overview

To help autonomous vehicles perform lane change maneuver wisely, we propose Cheetah, an innovative prediction-and-search framework that first predicts the near future trajectories for conventional vehicles and then searches for the optimal decisions from huge possible maneuver space. Figure 1 illustrates the workflow of the framework. Next we will briefly introduce each component separately.

Trajectory Prediction. In order to reduce the impact on conventional vehicles, the autonomous vehicle needs to predict the future trajectories of surrounding conventional vehicles. To this end, we develop an LSTM-based deep model, called GAS-LED, to predict the trajectories of z time steps ahead by utilizing historical trajectories in the past n time steps.

Instead of predicting the actual coordinates of each trajectory point directly, GAS-LED is designed to make dual prediction for lateral lane change behavior B_{lc} and distance of longitudinal motion behavior B_m of conventional vehicles at future time steps. For B_{lc}, the model estimates the probability of three behaviors and selects the behavior with the maximum probability as the predicted result. For B_m, the model outputs a longitudinal travel distance directly.

The most challenging part is how to maintain high accuracy in dynamic traffic environments, where the context of the autonomous vehicle keeps changing. To tackle the challenge, we adopt a parallel model architecture integrated with a global attention mechanism to improve the prediction accuracy. In addition, encoder and decoder state sharing is enabled to improve the convergence efficiency, so that the model can get up-to-date more quickly when the environment changes.

Maneuver Decision. Once the trajectory prediction model outputs the predicted maneuver sequences of conventional vehicles, Cheetah can start searching the optimal maneuver sequence for the autonomous vehicle and make decision on how to perform the maneuver. However, since the motion behaviors B_m is a continuous value, we need to discretize it first to make the search process feasible and then convert it back to continuous value during post-processing.

There are two goals during the search process. The first one is to maximize the average speed of the autonomous vehicle V_A, which is calculated as the longitudinal traveled distance divided by the time duration. The second one is to minimize the impact on surrounding conventional vehicles, which is characterized by an impact factor F_{im} to model the extent to which the conventional vehicles are affected by the maneuver of autonomous vehicle. We will discuss these two goals in more details in Section 4.

The challenge part of the maneuver sequence search is how to reduce search space and achieve high efficiency. To this end, we propose to use a maneuver tree structure to represent the entire search space. Using our adaptive beam search algorithm, the maneuver tree can be kept within a tractable size so the search can be performed efficiently.

3 TRAJECTORY PREDICTION

To reduce the impact of autonomous vehicles on conventional vehicles, i.e., F_{im}, it is necessary for autonomous vehicles to predict the future trajectories of other conventional vehicles. In Cheetah we propose a novel encoder-decoder network, called GAS-LED. GAS-LED divides the trajectory prediction task into two sub-tasks, namely lane change prediction and motion regression, which can assure the accuracy of lane-level prediction and the reliability of maneuver decision-making. Moreover, GAS-LED is designed to be trained efficiently with fast convergence so that the model can get up-to-date quickly when the environment changes dramatically.

3.1 Limitation of Existing Algorithms

Existing studies on trajectory prediction can be broadly classified into two categories, i.e., rule-based and learning-based trajectory prediction algorithms. In the sequel, we will briefly discuss the limitations of both types of methods w.r.t. to our objective.

Rule-based Trajectory Prediction. Rule-based trajectory prediction algorithms mainly apply the transportation rules to simulate traffic flow models. For example, a vehicle will move to the right lane when there is no vehicle on its right front, or it will speed up when there is no preceding vehicle. The simulated models can be applied to predict the future possible actions of vehicles according to the real-time road environment at each time step. The cellular automaton algorithms are capable of simulating the traffic flows [10, 24, 26], which complete the task efficiently in simple scenarios like one-way straight lane and coarse-grained scene. However, these algorithms ignore the historical trajectories of vehicles, which makes them difficult to perform long-term predictions. Besides, since these algorithms are designed for simple traffic conditions, the level of accuracy will be reduced significantly in more complex scenarios such as vehicles on a multi-lane road in our studied problem.

Learning-based Trajectory Prediction. In recent years, with a success achieved in applying RNN to model non-linear temporal dependencies in sequence learning tasks, there have been plenty of works [2, 3, 14, 23] utilizing RNN to predict the trajectories of vehicles. They have some common characteristics as follows:

- RNNs are adopted to extract long-term historical features.
- The scale of the networks is large, and there are many training parameters and hyperparameters involved.
- Single models output lateral and longitudinal information, which are continuous values in the Cartesian coordinate system.

These algorithms are able to predict long-term trajectories, and deal with the scenes with fine granularity and multi-interaction conditions based on the historical trajectory data of vehicles. However, the training process of the aforementioned network is usually time-consuming due to a large number of parameters and hyperparameters. This will hinder autonomous vehicles from updating model parameters promptly. Moreover, using a single model to predict both lateral and longitudinal information can reduce the accuracy of lane-level predictions, resulting in unreliable maneuver decisions.

In summary, a trajectory prediction model that is accurate, efficient and fine-grained is desired.

3.2 GAS-LED Model

We propose an LSTM encoder-decoder model with global attention and state sharing mechanism, called GAS-LED (Global Attention and State sharing based LSTM encoder-decoder), for trajectory prediction. More specifically, a global attention [25] mechanism is applied to assign different weights to the encoder state vectors for reflecting the importance of different time steps while avoiding complicating the model unduly. Besides, in order to improve the convergence efficiency of the model, we design an encoder and decoder state sharing mechanism to reduce the workload of calculation. Furthermore, we adopt a dual-model structure, i.e., two similar GAS-LED models operate in parallel to perform lane change classification and motion regression simultaneously. The two models share the same underlying structure, which are trained separately and optimized for their own prediction task to improve accuracy.

Let $\mathcal{T}^h$ denote a historical trajectory and $\mathcal{T}^f$ denote a predicted future trajectory. The length of $\mathcal{T}^h$ and $\mathcal{T}^f$ are indicated by n and z respectively, i.e., $\mathcal{T}^h = (p^{t-n+1}, p^{t-n+2}, \cdots, p^t)$ and $\mathcal{T}^f = (p^{t+1}, p^{t+2}, \cdots, p^{t+z})$, where t represents the current time step. At each time step, taking a historical trajectory $\mathcal{T}^h$ of the past n time steps as input, the GAS-LED model outputs the predicted trajectory $\mathcal{T}^f$ of the future z time steps. Next we give a detailed description of the GAS-LED model.

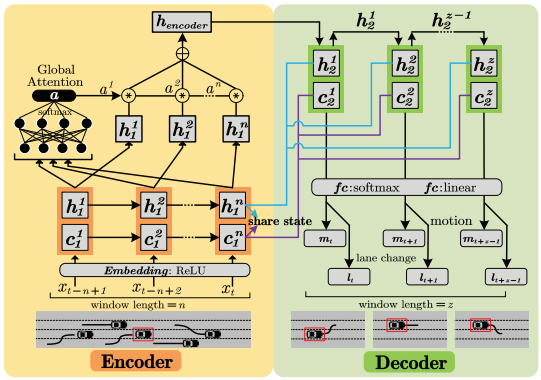

Figure 2: GAS-LED model structure

Input. The autonomous vehicle can get a set of 2-dimensional trajectory points from radar sensors including lateral lane number $p.L$ and longitudinal distance $p.D_{lon}$. GAS-LED needs to use a sequence of historical trajectory points to forecast a sequence of future trajectory points. The input features X of the models are $(x_{t-n+1}, x_{t-n+2}, \cdots, x_t)$ with the input window length n, where each x has 14 features: 2 for predicted vehicle (C_0) and 12 for its surrounding vehicles ($C_{1\sim6}$). These features are obtained from the historical trajectory $\mathcal{T}^h$ of conventional vehicles with length n. Predicted vehicle C_0 has two features: current lane number $p^t_{C_0}.L$ and longitudinal distance $p^t_{C_0}.D_{lon}$. For the surrounding vehicles $C_{1\sim6}$, we choose them based on the previous work [14], which have the most effect on a vehicle's motion. Each of them has two features: current lane number $p^t_{C_q}.L$ and relative longitudinal distance from the predicted vehicle $d(p^t_{C_0}, p^t_{C_q})$, where $q \in \{1,2,3,4,5,6\}$.

Output. In the task of lane-level trajectory prediction, we focus on which lane the vehicle is in, thus the model should output the probability of lane change behavior B_{lc}, and then select the behavior with the maximum probability as the predicted result. $\hat{l}_t$ denotes a 3-dimensional probability vector $[\hat{l}^1_t, \hat{l}^2_t, \hat{l}^3_t]$ at time t, where $\hat{l}^1$=$\mathcal{P}(Lt)$, $\hat{l}^2$=$\mathcal{P}(Rt)$ and $\hat{l}^3$=$\mathcal{P}(Nt)$. For the motion behavior B_m, the model should output the longitudinal travel distance $d(p^{t+1}_{C_0}, p^t_{C_0})$, denoted as $\hat{m}_t$.

For lane change prediction, the output of the model Y_l includes z 3-dimensional probability vectors, i.e., the classification probabilities of {Lt,Rt,Nt} in near future time steps. For motion prediction, the output of the model Y_m includes z continuous values (i.e., the longitude travel distance of the vehicle in the future time steps).

Prediction Objective. Obviously, the prediction of $\hat{l}$ is a classification task while the prediction of $\hat{m}$ is a regression task. In the

duration of interest $\mathbb{T}$, our model needs to minimize the loss function as follows:

$$Loss = \sum_{t\in\mathbb{T}}\sum_{i=1}^{3} -l_t^i \log(\hat{l}_t^i) + \frac{1}{|\mathbb{T}|}\sum_{t\in\mathbb{T}}(m_t - \hat{m}_t)^2 \quad (2)$$

where l_t^i and m_t denote the true values of the probability and distance respectively. $|\mathbb{T}|$ denotes the length of $\mathbb{T}$. This loss is the sum of Cross-entropy loss and Mean Squared error.

Model Structure. As shown in Figure 2, the historical information of the current vehicle and its surrounding vehicles is taken as the input of our GAS-LED model. Then, two similar models are applied in parallel to obtain multiple outputs simultaneously, i.e., the softmax activation for lane change classification prediction, and the linear activation for motion regression prediction. In the encoder part, the global attention assigns different weights to each time step, which can reflect the importance attached to the hidden states of different encoders [21]. Besides, the decoder uses the last hidden and cell state vector (h^n, c^n) of the encoder directly, as indicated by the blue and purple lines in Figure 2.

To be specific, in the encoder part, the input X is firstly embedded into $\tilde{X}$, i.e., $\tilde{X} = \phi_{eb}(X; W_{eb})$, where ϕ_{eb} is the embedding function with ReLU non-linerlity, W_{eb} denotes the embedding parameters.

Secondly, $\tilde{X}$ is defined as the input to encoder LSTM, and the LSTM outputs the hidden and cell states (h_1^t, c_1^t) at each time step, which is formulated as follows:

$$h_1^t, c_1^t = LSTM_1(h_1^{t-1}, c_1^{t-1}, \tilde{X}; W_{l1}) \quad (3)$$

where h_1^{t-1} and c_1^{t-1} are the hidden and cell state vectors at last time step respectively.

Thirdly, the global attention uses concatenated hidden state vectors to output the attention vector $\boldsymbol{a}$, which is calculated as follows:

$$\boldsymbol{a} = \phi_{at}(Concat(h_1^1, \cdots, h_1^n); W_{at}) \quad (4)$$

where ϕ_{at} is the attention function with softmax activation.

Finally, each element a^i in $\boldsymbol{a}$ represents the importance at each time step, and the output vector of the encoder part $h_{encoder}$ can be formulated as follows:

$$h_{encoder} = \sum_{i\in\{1,2,\cdots,n\}} a^i \cdot h_1^i \quad (5)$$

where $h_{encoder}$ extracts not only all the hidden states information but also the different degree at each time step.

In the decoder part, an LSTM is firstly used to decode the output vector from the encoder. However, this LSTM uses state sharing, which means that the state vectors of each time step equal to the last state vectors of the encoder, i.e., h_1^n,c_1^n. The decoder LSTM is calulated as follows:

$$h_2^t, c_2^t = \begin{cases} LSTM_2(h_1^n, c_1^n, h_{encoder}; W_{l2}) & ,t = 1, \\ LSTM_2(h_1^n, c_1^n, h_2^{t-1}; W_{l2}) & ,otherwise. \end{cases} \quad (6)$$

At the first time step, the input vector of the LSTM is the $h_{encoder}$, after which the input will be changed to the hidden state vector at the last time step.

Finally, the concatenated hidden state vectors are input to the output layer. Since we use two parallel models, and the calculations of these two models are same except for the output layer, we only distinguish the output calculation as follows:

$$\begin{aligned} Y_l &= \phi_{ol}(Concat(h_2^1, \cdots, h_2^z); W_{ol}), \\ Y_m &= \phi_{om}(Concat(h_2^1, \cdots, h_2^z); W_{om}) \end{aligned} \quad (7)$$

where ϕ_{ol} is the lane change output function with softmax activation, ϕ_{om} is the motion output function with linear activation.

To sum up, the lane change prediction outputs the specific lane change behavior in the future time steps, while the motion regression prediction outputs the continuous value for the longitudinal forward distance in the future time steps. The two outputs are simply combined to obtain the predicted trajectory points $(p.L, p.D_{lon})$ of future time steps for the vehicle C_0, as a result of the predicted future trajectory $\mathcal{T}_{C_0}^f$. The detailed implementations of the GAS-LED is described in Appendix.

4 MANEUVER DECISION

After having the predicted future trajectories of the surrounding vehicles, the autonomous vehicle needs to plan its maneuvers based on the prediction results. However, since the motion behavior B_m of the vehicle is a continuous value, performing a search against it directly will be extremely time-consuming. To speed up the search process, we develop a two-phase process for maneuver decision: *maneuver sequence search* and *maneuver process*. In the maneuver sequence search phase, we search for a discretized version of the optimal maneuver sequence of the autonomous vehicle. Specifically, a discretized maneuver is a pair $\tilde{M} = \langle B_{lc}, \tilde{B}_m \rangle$, where $\tilde{B}_m$ is a discretized version of B_m taking one of three behaviors: *speed-up* (Up), *speed-down* (Dw), and *maintain speed* (Mt). In the maneuver process phase, we generate the precise values for B_m based on $\tilde{B}_m$ obtained in the first phase. Since the logic of the maneuver process phase is easy to implement by adopting authoritative technology–Automotive Lane Change Aid (ALCA) [27], we mainly focus on maneuver sequence search in this section.

4.1 Problem Definition

Given a prediction horizon z, the goal of maneuver sequence search is to find a maneuver sequence $\tilde{\mathcal{M}} = \{\tilde{M}_t, \tilde{M}_{t+1}, ..., \tilde{M}_{t+z-1}\}$ of length z 1) to maximize the average speed of the autonomous vehicle; and 2) to minimize its impact on surrounding conventional vehicles. In the following, we first quantitatively define the impact factor used in search condition 2), and then formally define the search objective.

Impact Factor F_{im}**.** At time t, we assume the maneuver of the autonomous vehicle A can have impact on the set of conventional vehicles, denoted as $\mathbb{C}$, within a radius of R. Formally, the impact factor F_{im}^t is defined as the sum of the impact $F_{im}^{C_i,t}$ for each surrounding conventional vehicles $C_i \in \mathbb{C}$, i.e., $F_{im}^t = \sum F_{im}^{C_i,t}$. Specifically, we categorize the *impact situations* of maneuver within radius R into three types: queuing, jumping the queue, and crossing. Figure 3 illustrates the three scenarios, where the red shaded area, called conflicting zone, is the location that both vehicles plan to arrive at the next time step. Prior transportation studies have shown that crossing usually has the greatest impact, while queuing has the least [1, 17]. So we simply assign 1, 2 and 3 as the impact factors for queuing, jumping the queue and crossing. Next we present how to predict the three impact situations.

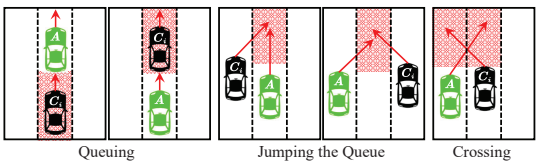

Figure 3: Three impact factors

Obviously, the impact situations depend on the future predicted trajectories of the surrounding vehicles and the future maneuver of the autonomous vehicle. A vehicle's status at time t can be described using a quadruple $(p^t.L, p^{t+1}.L, p^t.D_{lon}, p^{t+1}.D_{lon})$, which is used to describe its position at t and its immediate future position at $t+1$. Then the impact situations can be determined by checking if the autonomous vehicle and the conventional vehicle are on the same lane and their following distance is below the safe distance threshold. We use Algorithm 1 (in Appendix) to predict the type of impact situations and calculate the impact factor F_{im}^t.

Search Objective. We define the overall search objective as a linear combination of the above two search conditions. Specifically, at time step t, we aim to search for the maneuver sequence that maximizes the objective function defined below:

$$\arg\max_{\tilde{M}} \sum_{t\in[t,t+z]} V_A^t - F_{im}^t \tag{8}$$

It should be noted that we use the normalization for rescaling the range of features (V_A and F_{im}) to $[0, 1]$ before calculating the objective.

4.2 Searching Maneuver Sequences

In the following, we introduce the maneuver tree structure and present our adaptive beam search algorithm.

Maneuver Tree Structure. A maneuver tree is a tree-based data structure. The depth of the tree is z, and each non-leaf node has 9 child nodes (i.e., 3×3 pairs of discretized maneuver). Each node (except for the root) represents a kind of maneuver $\tilde{M}$ of the autonomous vehicle, and each edge holds two weights namely F_{im} (cost) and V_A (benefit) of choosing this edge. The root node is the current state while its child nodes represent the maneuvers.

Workflow Overview. At each time step, the maneuver tree will be reinitialized as a nine-complete tree with a fixed depth z, with the latest weights on each edge. First, based on the current trajectory point of the autonomous vehicle, some invalid nodes can be discarded based on the following rules:

- When the autonomous vehicle is in the rightmost lane, three direct child nodes (denoting the behavior of 'Change right') of the current node will be removed.
- When the autonomous vehicle is in the leftmost lane, three direct child nodes (representing the behavior of 'Change left') of the current node will be removed.

After deleting the invalid points, we use the adaptive beam search algorithm to search for the optimal path in the tree. Finally, the tree will be updated to prepare for the next search process.

Adaptive Beam Search Algorithm. For a search process, the search method is applied to search for a z-length path from the root node to the leaf node in the maneuver tree, while finding the search object. Existing search methods used a predetermined number of candidate nodes in each layer (called the width) and only those nodes are expanded next. i.e., Brute-force search (width=9), Greedy search (width=1) [12] and Beam search (1<width<9) [15]. The greater the width, the fewer nodes are pruned and the more time-consuming the search process is. In fact, the width may depend on the weight distribution of each layer, and thus should be predetermined more adaptively. For instance, in one layer, the weights are (0.01, 0.4, 0.5, 0.003), in another layer, the weights are (0.7, 0.6, 0.8, 0.9). If the algorithm fixes a small-sized width, it will ignore some valuable nodes (0.7,0.6), leading to a local optimum solution. Alternatively, if the algorithm fixes a large-sized width, the algorithm will consider excessive nodes (0.01,0.03). Therefore, the width in each layer depends on the different number of nodes with high weights (*a.k.a.* high-confidence).

Instead of fixing the search width, we adaptively select the candidate nodes with high confidence in each layer. To automatically select the candidate nodes, we define a threshold γ, which determines the minimum difference between high-confidence nodes and low-confidence nodes. Specifically, in i-th layer, We rank all the nodes based on their cumulative weights CW, i.e., $\sum(V_A^t - F_{im}^t)$ where $i \in [t, t+i]$. Then, we can select the high-confidence nodes (from $Top1$ to $TopK$) as the candidate nodes based on γ, i.e., $CW_{TopK} - CW_{TopK+1} \geq \gamma$. Thereafter, the nodes in $i+1$-th layer are the child nodes of the candidate nodes in i-th layer, while the non-candidate nodes are pruned to reduce search space.

The search procedure in our adaptive beam search algorithm is similar to the classical beam search algorithm [15]. At each time step, once reaching the search depth (i.e., z), the algorithm will evaluate the solutions found during search at various depths and return the best one (the one with the highest cumulative weights).

Example. As illustrated in Figure 4, nine maneuvers are shown on the right side of the figure, and the autonomous vehicle holds a maneuver tree. Each edge maintains a cumulative weight. In this case, the number of candidate nodes in each layer is different (4→6→7→5→5), following the threshold γ, and the non-candidate nodes are pruned accordingly. The search result is a path of length z ($z = 5$) with the maximum cumulative weights of $V_A - F_{im}$ (yellow line).

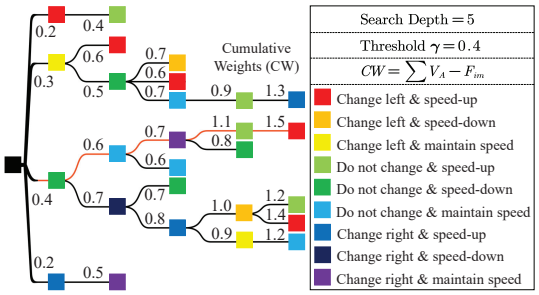

Figure 4: Adaptive beam search example

5 EXPERIMENTS

5.1 Experimental Settings

We implemented all models and algorithms in Python on Linux, and ran the experiments on a machine with an Intel(R) CPU i7-4770@3.4GHz and 32G RAM.

Datasets. As it requires interaction between the autonomous and conventional vehicles, most of the experiments were conducted in a simulated environment called **SIM**. **SIM** is generated using the microscopic traffic simulator [7, 34, 36], which simulates the traffic behaviors of 600 conventional vehicles on a cyclic six-lane road of length $3km$. We utilized Cheetah to control the autonomous vehicle to perform lane change maneuvers.

Furthermore, we tested the trajectory prediction model (GAS-LED) on a dataset constructed by merging two commonly-used

real-world datasets: NGSIM US-101 [9] and I-80 [16]. The merged dataset consists of real trajectories of conventional vehicles traveling on a $1.14km$-length freeway segment with six straight lanes. The original trajectories were captured at $10Hz$ (10 frames per second) over a period of 45 minutes. We preprocessed the dataset by 1) removing the frames with less than 10 vehicles, and 2) downsampling the datasets to a rate of $2Hz$ (setting the time step to 0.5s). After preprocessing, the merged dataset has $10,542$ frames with $9,864$ conventional vehicles. On average, there are 223 vehicles per frame, and the speed of vehicle is $33.4km/h$.

On both datasets, we set the traffic restrictions as $V_{min} = 0km/h$, $V_{max} = 115km/h$, $D_{ss} = 10m$ and $D_{lcs} = 10m$ according to [22].

Parameter Settings. Unless otherwise specified, we set the parameters in Cheetah throughout the experiments as follows. For the trajectory prediction phase, we set both the length of input historical trajectories and the length of predicted trajectories to 5 (i.e., $n = z = 5$), following the settings used in the previous work [13]. For the maneuver decision phase, we set $F_{im}^{C_i,t}$ to 1, 2 and 3 for queuing, jumping the queue and crossing, respectively. For the discretized motion behaviors $\tilde{B}_m$, we set the $speed-up$ acceleration to $1.2m/s^2$ and the $speed-down$ deceleration to $3m/s^2$, according to the rates recommended by [4]. Moreover, we set the adaptive beam search threshold as $\gamma = 0.4$, and the impact factor radius parameter as $R = 38m$, which are evaluated in Appendix.

Compared Methods. We compared Cheetah against several lane change algorithmas, including STNS [8], FLS [7], and SCMPC [5]. For the experiments on trajectory prediction, we further compared against the state-of-the-art trajectory prediction models including LSTM [3], S-LSTM [2], CS-LSTM [13], and ED-LSTM [23].

5.2 End-to-End Evaluation of Cheetah

In this section, we study the end-to-end performance of Cheetah by comparing it against several lane change algorithms (STNS, FLS, SCMPC). We conducted 100 tests on the simulation environment **SIM**–wherein each test the autonomous vehicle was initialized at a random position, and measured the effectiveness from both macroscopic and microscopic aspects:

• Macroscopic: we recorded the end-to-end time of all vehicles (the autonomous vehicle and all the 600 conventional vehicles) driving through $3km$.

• Microscopic: we recorded the average speed change rate of the conventional vehicles affected by (i.e., within radius $R = 38m$ of) the autonomous vehicle. The speed change rate is defined as $|V_{C_i}^{t+1} - V_{C_i}^{t}|/V_{C_i}^{t}(\%)$, where $V_{C_i}^{t}$ and $V_{C_i}^{t+1}$ are the speeds of a conventional vehicle C_i in two consecutive time steps. This metric directly reflects the extent of impact on the surrounding vehicles.

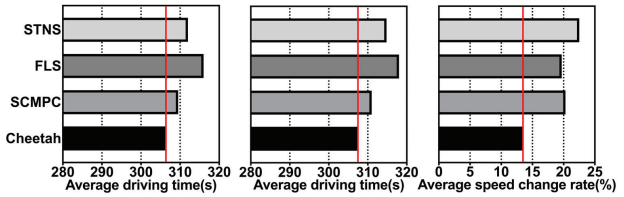

Figure 5: Effectiveness of Cheetah

Macroscopic. We report the average end-to-end driving time of the autonomous vehicle and the conventional vehicles in Figure 5a and Figure 5b. As shown, Cheetah achieves the shortest average driving time for both, which clearly demonstrates that Cheetah can maximize the average speed of the autonomous vehicle, while minimizing the impact on conventional vehicles.

Microscopic. We report the average speed change rate of the surrounding conventional vehicles in Figure 5c. We can see that Cheetah causes the least significant impact on the speeds of surrounding conventional vehicles, demonstrating the effectiveness of our framework from the perspective of microscopic traffic.

5.3 Evaluation of Trajectory Prediction

In this section, we take a break-down evaluation of our trajectory prediction model GAS-LED. We compared the trajectory prediction model in Cheetah with state-of-the-art trajectory prediction models (LSTM, S-LSTM, CS-LSTM, ED-LSTM). All compared models have the same input and output structures. All the experiments were done on the real dataset. The dataset was split into training and test sets by a ratio of 4 : 1, and the test errors were recorded.

Table 1: Effectiveness of Trajectory Prediction

(a) Lane Change Prediction

ACC(%)	prediction time step				
Methods	**1**	**2**	**3**	**4**	**5**
LSTM	91.2	89.7	88.5	88.1	87.6
S-LSTM	94.5	93.1	92.8	92.2	90.5
CS-LSTM	95.1	94.3	93.9	93.2	91.8
ED-LSTM	95.8	95.2	94.6	93.5	92.3
GAS-LED	**96.1**	**95.7**	**94.8**	**94.2**	**93.5**

(b) Motion Regression

MSE(ft)	prediction time step				
Methods	**1**	**2**	**3**	**4**	**5**
LSTM	0.60	0.89	1.39	1.58	1.87
S-LSTM	0.36	0.91	1.07	1.33	1.63
CS-LSTM	0.39	0.81	1.27	1.59	1.90
ED-LSTM	0.27	0.63	0.93	1.20	1.49
GAS-LED	**0.23**	**0.60**	**0.89**	**1.16**	**1.47**

Effectiveness of GAS-LED. We study the lane change prediction task and the motion regression task separately, and for both tasks we used historical trajectories of length 5 to predict trajectories of the next 5 time steps. We report the Accuracy (ACC) of the prediction task in Table 1a, and the Mean Squared error (MSE) of the motion regression task in Table 1b. As depicted, for both tasks, our proposed GAS-LED model outperforms all the other methods for all prediction time steps.

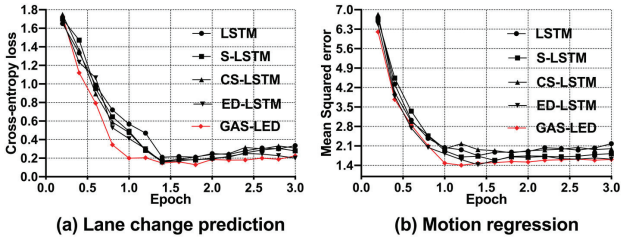

Figure 6: Efficiency of trajectory prediction

Efficiency of GAS-LED. To study the efficiency of the model, we recorded the curve of the training loss in the first three training

Table 2: Effectiveness of Updated Training Strategy

Strategies	Lane change ACC(%)	Motion MSE(ft)
CT	86.5	1.94
UD	**92.7**	**1.49**

epochs for all the prediction models. We choose the Cross-entropy loss and Mean Squared error as the loss metrics, and plot the loss curves in Figure 6a and Figure 6b. As shown, the loss of our GAS-LED model is minimized at around 1 epoch, while the other methods are minimized at around 1.5 epochs. This is because our model relies on the state sharing of encoder-decoder structure, thus reducing the training convergence time.

Necessity of a Continuously Updated Model. To demonstrate the necessity of a continuously updated model in maintaining a high trajectory prediction accuracy, we compared the prediction accuracy of GAS-LED under two different training strategies:

• Constant model (CT): We trained a single GAS-LED model using the entire training set, and used this model to do the prediction for the entire testing set.

• Updated model (UD): We further divided the training and testing dataset into time windows. We trained a separate GAS-LED model on each training window, and used it to do the prediction on the corresponding test window.

The results are shown in Table 2. We can see that the effect of the updated model is much better than that of the constant model. This clearly proves that the model needs to be updated in continuously changing environments in order to maintain accuracy.

5.4 Trajectory Prediction in Cheetah.

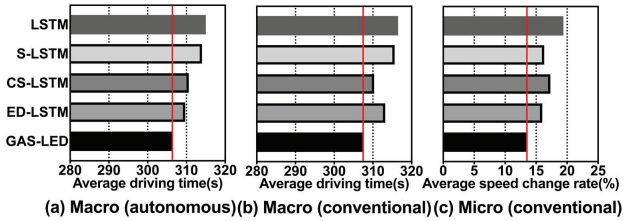

Figure 7: Effectiveness of prediction-and-search framework

We further study the effectiveness of GAS-LED by putting it in the Cheetah framework with maneuver search. We compared against other prediction models in terms of both macroscopic and microscopic effectiveness as in Section 5.2. The results are shown in Figure 7a ,7b and 7c. We can see that in the Cheetah framework with the maneuver search module, the proposed GAS-LED model outperforms all the competitive models, due to the synergy of our GAS-LED model and maneuver sequence search module. Other models pay more attention to location information rather than in the lane-change behaviors. This clearly shows that GAS-LED does not only outperforms other models in the stand-alone trajectory prediction task, but also is very effective in end-to-end lane change scheduling.

5.5 Evaluation of Maneuver Decision

In this section, we take a break-down evaluation of Cheetah's maneuver decision phase. We study maneuver decision by comparing different maneuver sequence search methods, i.e., Beam search (BM) [15] with fixed width = 4, Greedy search (GD) [12] with fixed width = 1, Brute-Force search (BF) with fixed width = 9, and our Adaptive Beam search (ABM). We conducted 100 tests on the simulation environment **SIM**, wherein each test the autonomous vehicle was initialized at a random position and drove $3km$.

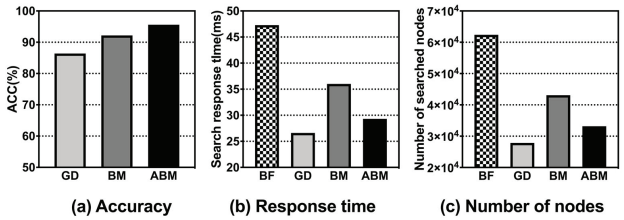

Figure 8: Effectiveness and efficiency of the search method

Search Effectiveness. To study the effectiveness of the adaptive beam search method adopted in Cheetah's maneuver decision phase, we compared the accuracy of the maneuver results of different search methods. As brute force search without any pruning (BF) is guaranteed to produce the globally optimal result, we defined the accuracy of {GD, BM, ABM} as the ratio that these methods will generate the same maneuver as BF. We report the accuracy results in Figure 8a. We can see that the accuracy of our ABM search method is noticeably better than other methods since we take adaptive nodes in each layer of the maneuver tree to search the optimal path. Our ABM improves the accuracy by $3.6 - 10\%$ compared to other methods, which evidence the effectiveness of Cheetah's search method.

Search Efficiency. We also compared the search response time and the number of searched nodes of different search methods in Figure 8b and 8c. As we can see, the ABM prunes $23 - 46\%$ nodes during the search process, which results in $18 - 38\%$ response time reduction for the beam search and the brute force search. The greedy search method takes the shortest response time since it only considers one candidate node in each layer. Therefore, it will mistakenly prune many high-confidence nodes, leading to a low accuracy for the searching process.

6 RELATED WORK

Lane change is one of the most conventional behaviors in autonomous driving. In this study, an autonomous vehicle is supposed to plan a series of lane change maneuvers. Existing lane change planing studies typically consider the driving safety and comfort of the vehicles [5–8, 35, 37, 38]. For example, [5] presents a novel control algorithm for lane change assistance and autonomous driving on highways based on Scenario Model Predictive Control (SCMPC). The basic idea is to account for the uncertainty in the traffic environment by a small number of future scenarios to perform safe lane change. [8] proposes a classical cellular automata model (called STNS) for planning the behaviors of vehicles, where a set of rules are applied to judgement the future lane change maneuvers. [35] presents a kinematics model for lane change, which can plan the trajectories based on the characteristics of polynomials. Besides, the infinite dynamic circles are applied to detect collision during lane change. [7] proposes a Freeway LaneSelection model (FLS), which will enable transportation professionals to more accurately model lane-changing behaviors on freeways. FLS algorithm consists of

choice of target lane and gap acceptance decisions, which aims to output the most accurate lane change maneuver.

Although the above lane change planning algorithms focus on the safety, comfort, and accuracy of the lane change, they ignore the potential impact of the lane change on other vehicles. Our study focuses on the impact of the lane change of autonomous vehicles, aiming to alleviate 'phantom' traffic jams. To the best of our knowledge, this paper pioneers to schedule the lane change maneuver by considering impact factors.

7 CONCLUSION

The development of vehicle automation technology has seen rapid progress in the last decade. However, changing lanes smartly is still a challenging problem for autonomous vehicles that lacks data-driven solutions. In this paper, we propose *Cheetah*, a prediction-and-search framework that helps the autonomous vehicle change lanes more wisely. In order to achieve high effectiveness and efficiency of autonomous driving, we addressed a few computational challenges by proposing a GAS-LED model to predict the future behaviors of neighboring vehicles based on their historical trajectories, and designing a maneuver-tree-based adaptive beam search algorithm to find the optimal maneuver sequence promptly. Extensive empirical studies based on both real and synthetic datasets also confirm the superiority of our proposed framework over the state-of-the-art approaches in terms of the average speed of autonomous vehicle, and its impact on surrounding vehicles.

ACKNOWLEDGMENTS

This work is supported by NSFC (No. 61802054, 61972069, 61836007, 61832017), Alibaba Innovation Research (AIR), scientific research projects of Quzhou Science and Technology Bureau, Zhejiang Province (No.2020D010).

REFERENCES

[1] Afzal Ahmed, Fatma Outay, Syeda Ofaq Raza Zaidi, Muhammad Adnan, and Dong Ngoduy. 2020. Examining queue-jumping phenomenon in heterogeneous traffic stream at signalized intersection using UAV-based data. *Personal and Ubiquitous Computing* (2020), 1–16.

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 961–971.

[3] Florent Altché and Arnaud de La Fortelle. 2017. An LSTM network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 353–359.

[4] PS Bokare and AK Maurya. 2017. Acceleration-deceleration behaviour of various vehicle types. *Transportation research procedia* 25 (2017), 4733–4749.

[5] Gianluca Cesari, Georg Schildbach, Ashwin Carvalho, and Francesco Borrelli. 2017. Scenario model predictive control for lane change assistance and autonomous driving on highways. *IEEE Intelligent transportation systems magazine* 9, 3 (2017), 23–35.

[6] M. Chen, Y. Zhao, Y. Liu, X. Yu, and K. Zheng. 2020. Modeling Spatial Trajectories with Attribute Representation Learning. *TKDE* (2020).

[7] Charisma Choudhury, Tomer Toledo, and Moshe Ben-Akiva. 2004. *NGSIM Freeway Lane Selection Model.* Technical Report. Federal Highway Administration.

[8] Debashish Chowdhury, Dietrich E Wolf, and Michael Schreckenberg. 1997. Particle hopping models for two-lane traffic with two kinds of vehicles: Effects of lane-changing rules. *Physica A: Statistical Mechanics and its Applications* 235, 3-4 (1997), 417–439.

[9] James Colyar and John Halkias. 2007. *Next Generation Simulation NGSIM US Highway 101 Dataset.* Technical Report. Federal Highway Administration.

[10] A Karim Daoudia and Najem Moussa. 2003. Numerical simulations of a three-lane traffic model using cellular automata. *Chinese journal of physics* 41, 6 (2003).

[11] LC Davis. 2004. Effect of adaptive cruise control systems on traffic flow. *Physical Review E* 69, 6 (2004), 066110.

[12] Fabrício Olivetti de França. 2018. A greedy search tree heuristic for symbolic regression. *Information Sciences* 442 (2018), 18–32.

[13] Nachiket Deo and Mohan M Trivedi. 2018. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1468–1476.

[14] Nachiket Deo and Mohan M Trivedi. 2018. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1179–1184.

[15] Markus Freitag and Yaser Al-Onaizan. 2017. Beam Search Strategies for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, 56–60.

[16] John Halkias and James Colyar. 2006. *Next Generation Simulation NGSIM Interstate 80 Freeway Dataset.* Technical Report. Federal Highway Administration.

[17] Robert Herman, Elliott W Montroll, Renfrey B Potts, and Richard W Rothery. 1959. Traffic dynamics: analysis of stability in car following. *Operations research* 7, 1 (1959), 86–106.

[18] SAE International. 2018. SAE International Releases Updated Visual Chart for Its "Levels of Driving Automation" Standard for Self-Driving Vehicles. https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-levels-of-driving-automation-standard-for-self-driving-vehicles

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Suzanne E Lee, Erik CB Olsen, Walter W Wierwille, et al. 2004. *A comprehensive examination of naturalistic lane-changes.* Technical Report. United States. National Highway Traffic Safety Administration.

[21] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[22] Kai Nagel, Dietrich E Wolf, Peter Wagner, and Patrice Simon. 1998. Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E* 58, 2 (1998), 1425.

[23] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. 2018. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1672–1678.

[24] Morten Monrad Pedersen and Peder Thusgaard Ruhoff. 2002. Entry ramps in the Nagel-Schreckenberg model. *Physical Review E* 65, 5 (2002), 056705.

[25] Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756* (2015).

[26] Marcus Rickert, Kai Nagel, Michael Schreckenberg, and Andreas Latour. 1996. Two lane traffic simulations using cellular automata. *Physica A: Statistical Mechanics and its Applications* 231, 4 (1996), 534–550.

[27] Kenneth Schofield. 2005. Automotive lane change aid. US Patent 6,882,287.

[28] David Schrank, Bill Eisele, and Tim Lomax. 2019. *2019 URBAN MOBILITY REPORT.* Technical Report. Texas A&M Transportation Institute.

[29] Tesla. 2020. MODEL S OWNER'S MANUAL. https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf

[30] Tomer Toledo and David Zohar. 2007. Modeling duration of lane changes. *Transportation Research Record* 1999, 1 (2007), 71–78.

[31] L. Wang and B. K. P. Horn. 2020. On the Stability Analysis of Mixed Traffic With Vehicles Under Car-Following and Bilateral Control. *IEEE Trans. Automat. Control* 65, 7 (2020), 3076–3083. https://doi.org/10.1109/TAC.2019.2945888

[32] M. Won, T. Park, and S. H. Son. 2017. Toward Mitigating Phantom Jam Using Vehicle-to-Vehicle Communication. *IEEE Transactions on Intelligent Transportation Systems* 18, 5 (2017), 1313–1324. https://doi.org/10.1109/TITS.2016.2605925

[33] RD Worrall and AGR Bullen. 1970. An empirical analysis of lane changing on multilane highways. *Highway Research Record* 303 (1970).

[34] Hwasoo Yeo, Alexander Skabardonis, John Halkias, James Colyar, and Vassili Alexiadis. 2008. Oversaturated freeway flow algorithm for use in next generation simulation. *Transportation Research Record* 2088, 1 (2008), 68–79.

[35] Feng You, Ronghui Zhang, Guo Lie, Haiwei Wang, Huiyin Wen, and Jianmin Xu. 2015. Trajectory planning and tracking control for autonomous lane change maneuver based on the cooperative vehicle infrastructure system. *Expert Systems with Applications* 42, 14 (2015), 5932–5946.

[36] Li Zhang, Shangshu Cai, Yunlong Zhang, and Min Zhang. 2010. Comparison of lane changing algorithms between NGSIM and CORSIM. In *2010 IEEE 71st Vehicular Technology Conference*. IEEE, 1–6.

[37] Yan Zhao, Shuo Shang, Yu Wang, Bolong Zheng, Quoc Viet Hung Nguyen, and Kai Zheng. 2018. REST: A Reference-based Framework for Spatio-temporal Trajectory Compression. In *SIGKDD*. 2797–2806.

[38] Kai Zheng, Yan Zhao, Defu Lian, Bolong Zheng, Guanfeng Liu, and Xiaofang Zhou. 2020. Reference-based Framework for Spatio-temporal Trajectory Compression and Query Processing. *TKDE* 32, 11 (2020), 12227–2240.

A APPENDIX

A.1 Notations

Table 3 shows the major notations used throughout the paper.

Table 3: Summary of Notations

Notation	Definition
C_i	A conventional vehicle C_i
A	An autonomous vehicle A
L_i	The i-th lane of the road
M_t	The lane change maneuver at time t
F_{im}	The impact factor of autonomous vehicle on conventional vehicles
$\mathbb{T}$	A time duration of interest
t	A time step $t \in \mathbb{T}$
V_A^t	The speed of vehicle A at time t
p_C^t	The vehicle's trajectory point at time t
D_{lon}	The vehicle longitudinal distance
$\mathcal{T}$	The vehicle trajectory

A.2 Time Step Setting

In this study, we treat the continuous time duration as a set of discrete time steps, i.e., $\mathbb{T} = \{1, 2, ..., t, t+1, ...\}$. It serves as the minimum frequency for autonomous vehicles to make lane changing decisions. Now the question is how to choose a suitable time granularity in real application scenarios? According to [30], the average duration of single-lane change is 4.6 seconds for a vehicle, which includes the transition time and the observation time required by the driver. In the field of microscopic traffic simulation [20, 33]. However, the transition time of lane change is often ignored and the duration of single-lane change is considered less than 1.2 seconds. Therefore, without loss of generality, the time granularity in this paper is set to 0.5 seconds, which means the time interval between two consecutive time steps is 0.5 seconds.

A.3 GAS-LED Model Implementations

In our implementation of GAS-LED trajectory prediction model, the encoder embedding layers contain 64 units with ReLU activation. The fully connected output layers for lane change and motion prediction are comprised of 3 units and 1 unit, respectively. The encoder and decoder LSTMs have 128 units, while the global attention involves two fully connected layers (activated by softmax), each with 64 units. We train the model using the Adam optimizer [19] for 15 epochs with a scheduled learning rate of 0.001 and a batch size of 64.

A.4 Impact Factor Procedure

In our maneuver decision phase, the impact factor F_{im}^t is defined as the sum of the impact $F_{im}^{C_i,t}$ for each surrounding conventional vehicles $C_i \in \mathbb{C}$, i.e., $F_{im}^t = \sum F_{im}^{C_i,t}$. To be specific, we categorize the impact situations of maneuver within radius R into three types: queuing, jumping the queue, and crossing. To predict the three impact situations, We define the vehicle's status at time t, i.e., a quadruple $(p^t.L, p^{t+1}.L, p^t.D_{lon}, p^{t+1}.D_{lon})$. Then the impact situations can be determined by using the statuses of the conventional and autonomous vehicles. Algorithm 1 depicts the algorithm for determining the type of impact situations and calculating the impact factor F_{im}^t.

Algorithm 1: Vehicle impact factor

Input: current time t, autonomous vehicle A, all conventional vehicles $\mathbb{C}$ in radius range R, $C_i \in \mathbb{C}$
Output: vehicle impact factor F_{im}^t

$F_{im}^t \leftarrow 0$;
foreach $C_i \in \mathbb{C}$ **do**
  $F_{im}^{C_i,t} \leftarrow 0$;
  if $(p_{C_i}^{t+1}.L) = (p_A^{t+1}.L)$ **then**
    if $|(p_{C_i}^{t+1}.D_{lon}) - (p_A^{t+1}.D_{lon})| < D_{ss}$ **then**
      if $(p_{C_i}^t.L) = (p_A^t.L)$ **then**
        $F_{im}^{C_i,t} \leftarrow 1$; // queuing
      else
        $F_{im}^{C_i,t} \leftarrow 2$; // jumping the queue
      end
    end
  else
    if $(p_{C_i}^{t+1}.L) = (p_A^t.L)$ ***and*** $(p_{C_i}^t.L) = (p_A^{t+1}.L)$ **then**
      if $|(p_{C_i}^{t+1}.D_{lon}) - (p_A^{t+1}.D_{lon})| < D_{lcs}$ ***or*** $|(p_{C_i}^t.D_{lon}) - (p_A^t.D_{lon})| < D_{lcs}$ **then**
        $F_{im}^{C_i,t} \leftarrow 3$; // crossing
      end
    end
  end
  $F_{im}^t \leftarrow F_{im}^t + F_{im}^{C_i,t}$;
end
return F_{im}^t;

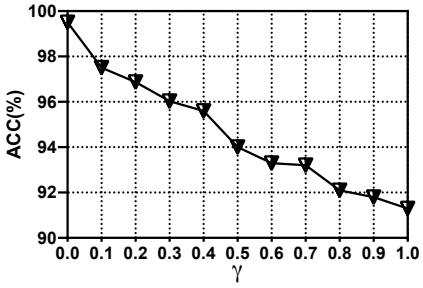

Figure 9: Accuracy of threshold γ of adaptive beam search

A.5 Effect of Hyperparameters

Next we study the effect of the hyperparameters in Cheetah.

Effect of Adaptive Beam search Threshold γ. We conducted a grid search for the threshold γ used in our adaptive beam search from 0 to 1 with stride 0.1, and measured the accuracy and response time of maneuver sequence search. The results are depicted in Figure 9 and Figure 10. We can see that when γ is greater than 0.4, the accuracy begin to decline rapidly while the response time tends to stabilize. Due to the above observation, We choose $\gamma = 1.4$ to balance accuracy and efficiency.

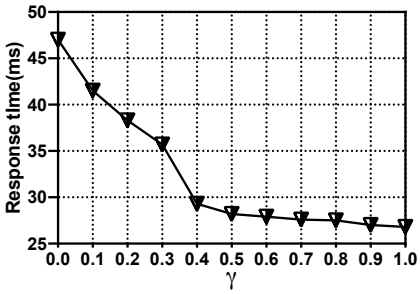

Figure 10: Response time of threshold γ of adaptive beam search

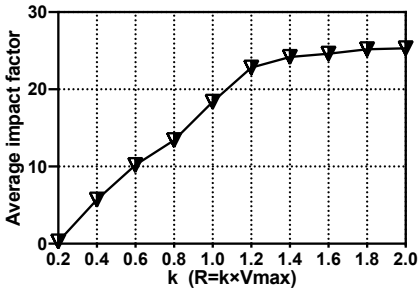

Figure 11: Effect of radius parameter R

Effect of Radius Parameter R of Impact Factor F_{im}. The radius of the impact factor determines the size of the nearby environment that the autonomous vehicle needs to consider. An overly large radius will cause calculation redundancy; conversely, an overly small radius will ignore important nearby vehicles. And intuitively, the radius is proportional to the maximal speed for the road. Therefore, the radius is formalized as: $R = k \times V_{max}$. We tested the parameter k at different V_{max} values, i.e., highway=32m/s, urban=15m/s, 20m/s and downtown=5m/s, so as to calculate the average impact factor under different R. We took the average value of the impact factor under each candidate value of V_{max} to compare the different values of k. As shown in the Figure 11, when k equals to 1.2, the increasing trend of the impact factor slows down, which means the impact factor tends to stabilize. Thus, $k = 1.2$ is taken as the default. Correspondingly, we set the radius parameter in the experiments as $R = 38m$ w.r.t. V_{max}=$32m/s$ ($115km/h$).