

Graph-based Prediction and Planning Policy Network (GP3Net) for scalable self-driving in dynamic environments using Deep Reinforcement Learning

Jayabrata Chowdhury¹*, Venkataramanan Shivaraman²*, Suresh Sundaram¹, P B Sujit²

¹Indian Institute of Science, Bengaluru ²Indian Institute of Science Education and Research, Bhopal
jayabratac@iisc.ac.in, vshivaraman18@gmail.com, vssuresh@iisc.ac.in, sujit@iiserb.ac.in

Abstract

Recent advancements in motion planning for Autonomous Vehicles (AVs) show great promise in using expert driver behaviors in non-stationary driving environments. However, learning only through expert drivers needs more generalizability to recover from domain shifts and near-failure scenarios due to the dynamic behavior of traffic participants and weather conditions. A deep Graph-based Prediction and Planning Policy Network (GP3Net) framework is proposed for non-stationary environments that encodes the interactions between traffic participants with contextual information and provides a decision for safe maneuver for AV. A spatio-temporal graph models the interactions between traffic participants for predicting the future trajectories of those participants. The predicted trajectories are utilized to generate a future occupancy map around the AV with uncertainties embedded to anticipate the evolving non-stationary driving environments. Then the contextual information and future occupancy maps are input to the policy network of the GP3Net framework and trained using Proximal Policy Optimization (PPO) algorithm. The proposed GP3Net performance is evaluated on standard CARLA benchmarking scenarios with domain shifts of traffic patterns (urban, highway, and mixed). The results show that the GP3Net outperforms previous state-of-the-art imitation learning-based planning models for different towns. Further, in unseen new weather conditions, GP3Net completes the desired route with fewer traffic infractions. Finally, the results emphasize the advantage of including the prediction module to enhance safety measures in non-stationary environments.

Introduction

Motion planning for AVs still has a long way to go due to the complexity of real-world driving environments, ensuring the safety and comfort of everyone. The AV must be able to carefully plan its maneuvers in urban and highway environments with varying traffic dynamics set by cars, bikers, pedestrians, and weather conditions. A typical intersection scenario in an urban environment is shown in Fig.1. Various traffic participants' different intentions and final goals (as shown by arrows in Fig.1) make it challenging for AV to safely and comfortably move to its destination. Variable

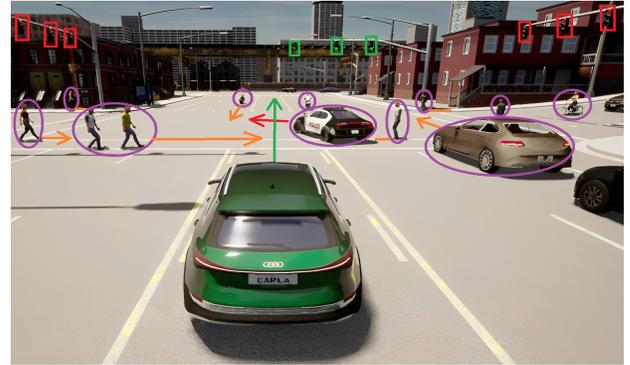


Figure 1: A typical intersection scenario with traffic lights, pedestrians, and vehicles. The scenario has pedestrians and vehicles moving on the desired path of AV. The figure has colour coded markers to show the context.

speeds of different traffic participants also make it difficult for AV to drive safely. A research finding on the United States (Choi 2010) shows that wrong estimation of other traffic participants' speed is responsible for 8.4% of critical reasons for crashes. Also, another problem of driving in an unstructured environment is that traffic pattern varies with the cooperative/noncooperative behaviors of other traffic participants. By utilizing communication channels as described in (Bazzi et al. 2021; Cui et al. 2022), it becomes possible to determine the intentions and objectives of fellow traffic participants. This information simplifies the process for AVs to make informed decisions regarding their actions. However, vehicles use different communication protocols, and ensuring reliable real-time communications between all traffic participants is challenging. Hence, another way is to model the interactions of traffic participants through time and predict their future trajectories to understand their behaviors. An AV can make a safe and efficient decision in a non-stationary environment if it can model the interactions better and predict the future trajectories of other surrounding vehicles. Depending on the future trajectories of others', an AV can decide on a safe planning maneuver. Hence, there is a need to develop a motion planning algorithm that can handle non-stationary environments using the intention understanding from the predicted trajectories of other traffic

*These authors contributed equally.

participants.

Existing state-of-the-art methods can be broadly classified into rule-based, imitation learning-based, and reinforcement learning-based works. Recently (Aksjonov and Kyrki 2021) uses a rule-based decision-making system for navigating road intersections. However, due to the non-stationary nature of the environment, rules vary for each situation, and attempting to create generalized rules across diverse environments suffers from scalability issues. On the other hand, some models mimicking expert drivers, called Imitation Learning (IL) based motion planner, has been proposed in (Bansal, Krizhevsky, and Ogale 2018; Rhinehart, McAllister, and Levine 2020; Codevilla et al. 2018; Cai et al. 2020, 2021). The driving model in (Bansal, Krizhevsky, and Ogale 2018) learns driving decisions from expert human driver demonstrations. The work in (Rhinehart, McAllister, and Levine 2020) uses expert driving data collected from CARLA (Dosovitskiy et al. 2017) simulator to learn expert behaviors. An end-to-end motion planning algorithm has been learned with IL in (Cai et al. 2020; Codevilla et al. 2018). However, the IL method suffers from a data distribution shift from training scenarios to unseen evaluation scenarios as (Bansal, Krizhevsky, and Ogale 2018) did not perform well in unknown highly interactive lane change scenarios. In (Cai et al. 2021), a graph-based attention network has been used to model participant interactions, and a driver model learns to imitate the expert with IL. In another work (Teng et al. 2023), a Bird’s Eye View (BEV) representation has been generated from the front camera view to use with IL for decision-making. The expert bias and distribution shift problems heavily influence the aforementioned IL-based works’ performance. Since there are no near-collision scenarios in expert driving, it will be difficult for IL-based methods to perform in the real environment.

Another popular approach to overcome the expert-based IL method is Reinforcement Learning (RL). In RL, the agent learns through their experience of completing the task. Recently works (Chen, Yuan, and Tomizuka 2019; Chen, Li, and Tomizuka 2021; Ye et al. 2020; Tang et al. 2022; Alizadeh et al. 2019; Naveed, Qiao, and Dolan 2021) employed RL for motion planning and maneuver decision-making for autonomous driving. In (Chen, Yuan, and Tomizuka 2019; Chen, Li, and Tomizuka 2021), a BEV image-based representation has been used to represent the AV’s surrounding environment. An RL agent learns to make maneuver decisions with this input representation. The works in (Ye et al. 2020; Tang et al. 2022) learn lane-changing maneuvers in simulated handcrafted driving scenarios through policy optimization methods ((Schulman et al. 2017; Haarnoja et al. 2018)). Recently, a work (Chen et al. 2020) used conditional Deep Q Network (CDQN) for stable driving decisions in the CARLA environment. However, this work did not include other vehicles, pedestrians, or traffic lights to model spatio-temporal interactions with AV. The works mentioned above need to recognize the intent of surrounding traffic participants. In (Huang et al. 2019), the intent of surrounding vehicles is calculated probabilistically to solve motion planning problems. However, the intent is modeled for a fixed number of vehicles without pedestrians and evaluated in a hand-

crafted driving scenario. The intentions of varying numbers of surrounding vehicles should be inferred to understand how the non-stationary environment can evolve. Hence, a predictive motion planning algorithm is required to model the spatio-temporal interactions for trajectory prediction and make a safe and efficient maneuver decision.

The proposed deep Graph-based Prediction and Planning Policy Network (GP3Net) framework finds a way to handle distribution shifts, model the interactions to predict trajectories and plan safe maneuvers. The GP3Net’s benefits are as follows:

1. A deep spatio-temporal graphical model encodes interaction to depict the behaviors of traffic participants with AV. These modeled interactions are passed through the trajectory prediction module to provide predicted future trajectories with associated prediction uncertainties.
2. The future mask generation module generates a BEV occupancy map with the prediction uncertainties, capturing the intentions of other traffic participants and probable evolution of a non-stationary environment in the future.
3. The AV neural architecture (state-encoder plus policy network) encodes past BEV masks and predicted future BEV masks together to give the final state.
4. The AV neural architecture outputs the values of acceleration and steering directly. The whole AV neural architecture is updated end-to-end with the RL algorithm.
5. The learned GP3Net driving agent outperforms the previous model DiGNet (Cai et al. 2021) and HILL (Teng et al. 2023) in CARLA Leaderboard and No-Crash motion planning benchmark in different towns and weathers. A Qualitative analysis highlights the importance of the graph-based trajectory prediction module.

Related Work

Trajectory prediction

Trajectory prediction is a crucial part of autonomous driving involving spatiotemporal interactions between traffic participants. In (Ma et al. 2019; Xu, Yang, and Du 2020), Long-Short Term Memory (LSTM) network has been used for modeling. However, these works need to model the influence of one traffic participant on another. Recently, (Shi et al. 2022) proposes a tree-based attention method. Also, in (Li and Zhu 2021), spatio-temporal modeling has been by a graphical model. However, these approaches needed to provide uncertainties in trajectory prediction. In (Ivanovic and Pavone 2019), a spatio-temporal dynamic graph models interactions, and LSTM networks model the state encoding of graph nodes. A Conditional Variational Auto-Encoder (C-VAE) learns the trajectory distribution. Samples from this probability distribution are taken for Gaussian Mixture Model (GMM)-based multimodal trajectory prediction. Trajectron predicts future trajectories with prediction uncertainties. GP3Net uses Trajectron architecture for trajectory prediction of surrounding traffic participants.

End-to-end and semantic-based motion planning

Many works directly use sensors (camera, LIDAR) to understand the surrounding environment and make decisions. In

(Codevilla et al. 2018), a directional high-level command is used to learn low-level steering and throttle values. The following work (Codevilla et al. 2019) showed the limitations of IL for autonomous driving. Recently (Teng et al. 2023) uses a front camera as a vision sensor and converts it to BEV representation of the driving scene. However, this work does not model rear-end side of the AV in the BEV. The rear-end side is crucial for AV because 64.2% of AV crashes happen from the rear end as reported in (Petrović, Mijailović, and Pešić 2020). GP3Net uses the semantic BEV representation for deciding safe maneuvers.

The GP3Net Framework

This section introduces the GP3Net motion planning framework that incorporates future trajectory predictions generated by a trajectory prediction module into Deep Reinforcement Learning-based motion planning. The subsections describe the self-driving task as a Partially Observed Markov Decision Process (POMDP). The primary input for the GP3Net consists of past and predicted future BEV occupancy maps containing information about road layout, vehicles, pedestrians, and traffic lights. The predicted future occupancy maps of vehicles are generated using a spatio-temporal graph-based trajectory prediction module based on (Ivanovic and Pavone 2019). Fig. 2 is a flowchart showing the pipeline between input processing, future BEV occupancy map generation, and output control actions.

POMDP Formulation

POMDP is defined for this task as a 6-tuple (S, A, T, R, O, γ) . Here, S is a set of possible states in the environment, A is the action space that the agent/AV can take in the environment, T is the transition probability $T(s'|s, a)$ that represents the probability of AV moving to state s' after taking action a from state s . R is the reward function $R : S \times A \rightarrow \mathbb{R}$ for the AV, O is a set of observations the AV can get, and $\gamma \in (0, 1]$ is the discount factor. The goal is to learn a policy π which can maximize the expected cumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where r_t is the reward obtained in the time step t . Following subsections define the observation space, action space, and reward function for the driving task in CARLA simulator.

Input Preprocessing and Observation Space Design

The observation space is $\mathbf{O} = [O_c; O_p; O_f; O_m]$. This observation vector \mathbf{O} for RL-based AV consists of several components of particular categories. The observations O_c , O_p and O_m , except O_f , are directly obtained from environment. The observations \hat{O}_c , \hat{O}_p and \hat{O}_f are rasterized, segmented local BEV image/mask centered around the ego vehicle. The O_c contains segmented BEV masks of the road layout, the mission route, and lane boundaries for contextual localization at time-step t . Observation O_p contains past configurations masks of the surrounding vehicles, pedestrians, traffic lights, and stop signs for k time-steps in the past, each 4 time-steps apart. Observation $O_m = [\tau, b, \omega, \dot{\mathbf{x}}]$ is a simple vector containing the AV's odometric measurements such as throttle τ , braking b , steering ω , and velocity $\dot{\mathbf{x}} = (\dot{x}, \dot{y})$ at time-step t

as shown in Fig. 2. The vector O_f contains the predicted occupancy maps (added as masks) of surrounding vehicles for l time-steps in the future, spaced 4 time-steps apart. These future occupancy maps in O_f are accurately generated by a spatio-temporal graph-based trajectory prediction module. This trajectory prediction model takes in the observed past *trajectories* of surrounding vehicles/pedestrians and models their interactions as dynamic spatiotemporal graphs to accurately predict future trajectories. The future occupancy maps are generated using the future trajectories with respect to our AV.

Graph-based Trajectory Prediction Module: The trajectory prediction module takes past kinematics information of the surrounding vehicles as input to generate a distribution of future trajectories for all agents present in a scene. The most probable future trajectory is then selected for generating the future maps. This module doesn't assume a fixed number of vehicles in the sensor range of the AV. Let $\mathbf{x}_{(t-k, t)}^i = [(x, y)_{t-k}, (x, y)_{t-(k-1)}, \dots, (x, y)_t]$ are the spatial coordinates of agent i from timestep $t - k$ to t . The objective is first to predict the future trajectories $\mathbf{x}_{(t+1, t+T)}^1, \mathbf{x}_{(t+1, t+T)}^2, \dots, \mathbf{x}_{(t+1, t+T)}^N$ for T timesteps for all N agents which were present in the scene in that time window. Formally, the input for the model is $\mathbf{x} = [\mathbf{x}_{1:t_{obs}}^{1, \dots, N}, \dot{\mathbf{x}}_{1:t_{obs}}^{1, \dots, N}, \ddot{\mathbf{x}}_{1:t_{obs}}^{1, \dots, N}]$ and the output $\mathbf{y} = \dot{\mathbf{x}}_{(t_{obs}+1, t_{obs}+T)}^{1, \dots, N}$. The module contains a dynamic spatio-temporal graph model, deep generative CVAE with LSTM-based encoder/decoder, and GMM. This architecture directly models the velocity rather than the trajectories. When the future velocities are obtained, they are numerically integrated to produce future trajectories using single-integrator dynamics, as given below. Here, δt is the difference of consecutive timesteps.

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i + \mathbf{y}_t^i \cdot \delta t \quad (1)$$

The dynamic spatiotemporal graph representation effectively models dynamic interactions between the agents present in the scene. The graphs are created based on the spatial proximity of the agents. The CVAE framework attempts to learn the probability distribution function $p(\mathbf{y}|\mathbf{x})$ as given below, where z is a discrete latent variable.

$$p(\mathbf{y}|\mathbf{x}) = \sum_{z \in Z} p_{\psi}(\mathbf{y}|\mathbf{x}, z) p_{\phi}(z|\mathbf{x}) \quad (2)$$

The encoder architecture consists of a node history encoder, a node future encoder (only during training), and an edge encoder.

$$e_{t, node}^i = LSTM(e_{t-1, node}^i, \mathbf{x}_t^i; W_{NE}) \quad (3)$$

Both graph node history and future encoders have LSTM networks, as shown in Equ.3 for encoding node data and temporal relations between them.

$$f_t^i = \left[\mathbf{x}_t^i; \sum_{j \in N(i)} \mathbf{x}_t^j \right]; a_t^i = LSTM(a_{t-1}^i, f_t^i; W_{EE}) \quad (4)$$

The edge encoder also uses LSTM network as given in Equ.4 and a modulating filter to encode the influence of other

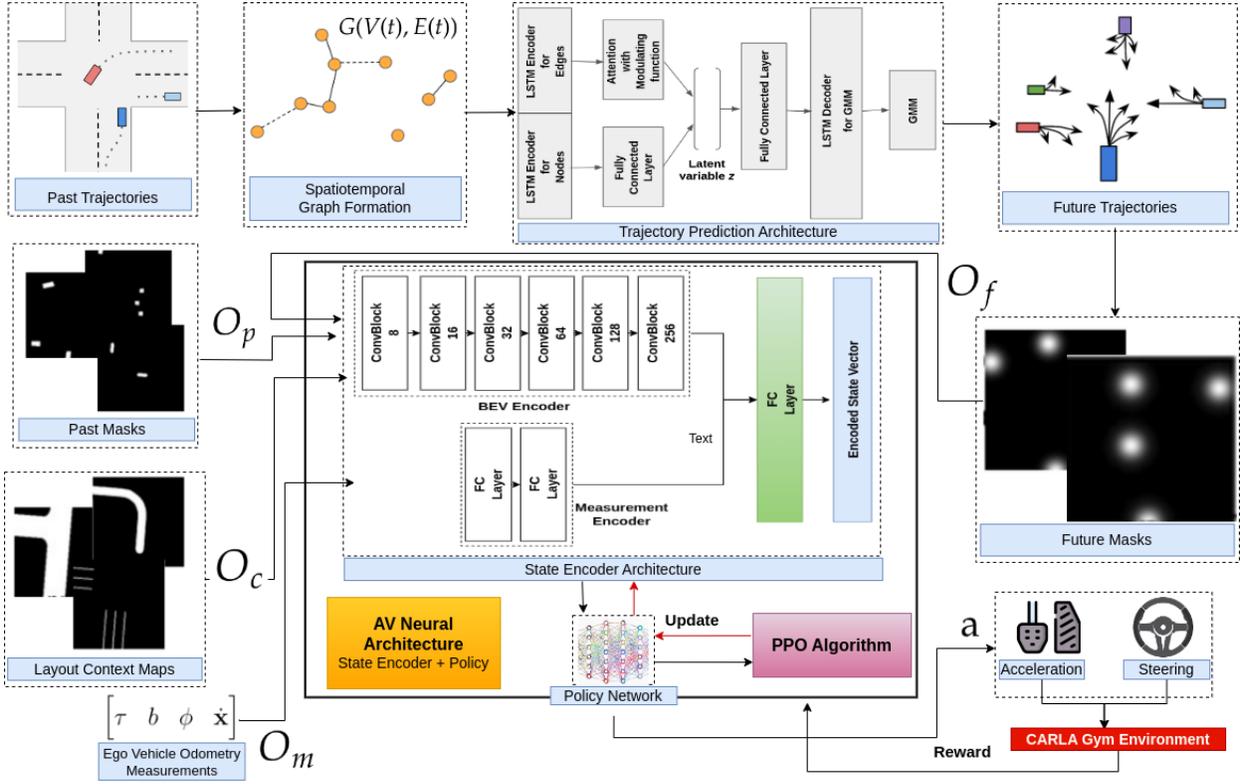


Figure 2: Schematic diagram of the GP3Net framework. The diagram shows the pipeline from input observations \mathbf{O} to control outputs a and ω , highlighting the future occupancy map generation by the trajectory prediction module, the state-encoder module, the PPO algorithm that optimizes the AV’s state-encoder and policy network.

agents with smooth edge additions and removals if high-frequency changes occur in the graph. In Equ.4, $[a; b]$ is concatenation, $N(i)$: set of neighbors of agent i . The latent variable z is then sampled and fed to the decoder consisting of an LSTM network and GMMs. The outputs for the decoder’s LSTM network are the GMMs’ parameters. The GMMs are then used to produce the future velocities and trajectories. The future masks containing future occupancy maps for the surrounding vehicles are generated after obtaining the predicted future trajectories for all the vehicles inside AV’s sensor range.

Future Mask Generation: After getting the predicted future trajectories $\mathbf{x}_{t+1, t+T}^{1, \dots, N}$, the cartesian coordinates $\mathbf{x}_{t+1, t+T}^i$ are converted to pixel coordinates $\mathbf{p}_{t+1, t+T}^i$ centered around the AV for each vehicle/agent i . The existence of a vehicle in the future mask is denoted by a 2D symmetric Gaussian patch as shown in Fig.2. The pixel coordinates \mathbf{p} serve as the mean of the 2D symmetric Gaussian function. The symmetric Gaussian approach helps us account for the future prediction uncertainties of the vehicles, accounting for various factors such as sensor noise and occlusion. Equation of future vehicle patch is given by:

$$g(x, y) = \exp\left(-\frac{((x - x_o)^2 + (y - y_o)^2)}{2\sigma^2}\right) \quad (5)$$

In the above equation, σ is the standard deviation of the

Gaussian, x_o and y_o are the mean of the 2D Gaussian. For each timestep t_f in future prediction horizon T of the trajectory prediction model, a future mask is generated using pixel coordinates $\mathbf{p}_{t_f}^{1, \dots, N}$ of all N vehicles in the method mentioned above. Therefore, O_f consists of l future masks. The observations are taken 4 timesteps apart instead of 1, and the inputs to the trajectory prediction module from timesteps $t - k$ to t are passed in 4 timestep intervals but processed as though inputs are of consecutive timesteps. This is further explained in the experiments section of the paper.

Final State Encoder: Once O_c , O_p , O_f , and O_m are obtained, they are passed through a state-encoder (first part of AV Neural Architecture) which results in a column vector. The architecture for the state encoder, as shown in Fig. 2, has two parts. The first part processes the BEV image masks O_c , O_p and O_f using a Convolutional Neural Network (CNN) as input channels and a simple Fully Connected (FC) layer. The measurement vector O_m is encoded using two FC layers and concatenated with the output of the BEV encoder, passed through an FC layer, finally giving the final state representation for the RL algorithm.

Action Space

The CARLA-Gym simulator takes three values for actions: throttle τ , brake b , and steering ω . However, to train the RL algorithm, action space is defined as a 2D continuous space

$\mathbf{a} = [\max\{\tau, b\} \cdot \text{sgn}(\tau - b), \omega]$, where $\text{sgn}(\cdot)$ is signum function. The first expression in \mathbf{a} is acceleration whose value ranges from -1 (maximum braking) to 1 (maximum throttle), and the steering value goes from -1 (maximum left steering) to 1 (maximum right steering). If the acceleration value is positive, it is passed to throttle τ , and brake value b is zero. If the acceleration value is negative, the magnitude of the acceleration is passed as brake b value, and throttle τ is zero. To select actions from this continuous space, the policy network π_θ is trained to output parameters α and β for a Beta distribution $\mathcal{B}(\alpha, \beta)$ for an improved policy gradient method (Chou, Maturana, and Scherer 2017). The throttle and steering actions are sampled from this Beta distribution. This work uses separate Beta distributions for throttle and steering.

Reward Structure

The reward structure of RL is essential to train agents/AV to perform specific tasks. The reward is a superposition of different components, as given below.

$$R = r_{route} + r_{halt} + r_{vel} + r_{pos} + r_{hd} + r_{act} \quad (6)$$

where the AV is penalized, and the episode is terminated whenever the vehicle collides with anything, runs a traffic light or a stop sign, routes deviation, or blocks. $r_{route} = -1$ if route deviation $\delta \geq 3.5m$; $r_{halt} = -1$ if AV's velocity $v \leq 0.1m/s$; $r_{vel} = 1 - |v - v_{assigned}|/v_{max}$ where v_{max} is maximum velocity allowed and $v_{assigned}$ is assigned by the simulator; $r_{pos} = -0.5x$ where x is the distance from the AV's center and the center of the desired route; $r_{hd} = -\Delta_h$ absolute difference between heading of AV and route; $r_{act} = -0.1$ if $|\omega_t - \omega_{t-1}| > 0.01$.

PPO Algorithm

The AV's policy network π (second part of AV neural architecture) learns using a state-of-the-art policy gradient-based RL algorithm called Proximal Policy Optimization (PPO) (Schulman et al. 2017). The usual form of loss function used in policy gradient methods is shown below.

$$L_{pg} = \mathbb{E}_\pi \left[\hat{A}_t \log \pi(a_t | s_t, \theta) \right] \quad (7)$$

Here, $\pi_\theta(a_t | s_t, \theta)$ is a stochastic policy with θ as parameter. \hat{A}_t is an estimator of the advantage function at time t , and \mathbb{E}_π represents the empirical average of a finite batch of samples.

However, updating the policy network π_θ for optimizing the objective loss function shown in Equ.8 can lead to large policy updates, often resulting in instability and bad policy updates. A new objective function called the clipped surrogate objective, which places a constraint on policy updates, is proposed in PPO paper. This makes the training process much more stable and reliable when updated over multiple epochs of gradient ascent. Also, a Generalized Advantage Estimate (GAE) and adding an entropy bonus with their new clipped surrogate objective function is used, as shown in the

following equation for PPO training of the GP3Net model.

$$\mathcal{L}_{ppo} = \mathbb{E} \left[\min(\tilde{r}(\theta) \hat{A}, \text{clip}(\tilde{r}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}), -c_1(V_\phi(s, a) - V_{target}) - c_2 H(s, \pi_\theta) \right] \quad (8)$$

Here in Equ.8, ϵ , c_1 and c_2 are tunable hyper-parameters and $H(s, \pi_\theta)$ denotes the entropy bonus. The generalized advantage estimate makes use of a learned state function $V_\phi(s, a)$ for computing the advantage function \hat{A} . The entropy bonus encourages the AV to explore adequately. Most importantly, during update, the whole AV Neural Architecture gets updated.

Experiments

Training Setup

This subsection describes the trajectory prediction module's training setup and the RL-based motion planning training.

Trajectory Prediction Module Training First, trajectory data of surrounding vehicles are collected as observed by the CARLA simulator's *roaming agent* in different towns for training the trajectory prediction module. The trajectory data contains rectangular coordinates $\mathbf{x}_t = (x_t, y_t)$ of all vehicles at time-steps $t, t + 4\delta t, t + 8\delta t \dots$ with each time-step spaced $\delta t = 0.1$ seconds apart in simulation time. The trajectory prediction module is provided with 3.2 seconds of observed trajectories as input, divided into eight discrete time-steps ($t - 29\delta t, t - 25\delta t, \dots, t - \delta t$), and the model predicts the trajectories up to 2.8 seconds, divided into seven time-steps ($t + 4\delta t, t + 8\delta t, \dots, t + 28\delta t$).

The training starts by initializing the weights and setting the batch size to 64 and the learning rate to 0.001 with a decay rate of 0.9999 to improve convergence. Adam optimizer updates the weight parameters of the model. Trajectory prediction models are typically benchmarked by comparing their performance against actual or simulated ground truth trajectories to evaluate accuracy and reliability. Here Mean Squared Error (MSE) metric evaluates the performance and accuracy of the model. The MSE is the mean L_2 error between the ground truth trajectories and predicted trajectories. The model was trained over 2000 steps and the MSE was evaluated intermittently over a validation set as shown in (Ivanovic and Pavone 2019).

RL Simulation Environment Setup

The CARLA simulator is employed as a gym environment for training the RL algorithm and evaluating proposed RL-based motion planning. The training was conducted in parallel on six distinct town maps, namely Town 1 to Town 6, employing a vectorized environment configuration. This training approach on multiple maps simultaneously enhanced the replay buffer's diversity and improved the training process's efficiency. During the rollout phase, the AV is provided with a mission route at the beginning of each episode. This route is generated by randomly selecting a source and destination point and connecting them using a path search algorithm like A^* . The AV aims to successfully navigate the environment map and complete the assigned mission route. The episode

would terminate if any of the following conditions were met: 1) collision with a vehicle, pedestrian, or obstacle in the map layout; 2) deviation from the given route; 3) traffic stagnation/halt and 4) violation of traffic lights and stop signs. Upon reaching the destination point, the AV was assigned a new mission route, continuing the rollout phase. The training and experiments were run in parallel on a Ubuntu 20.04 machine with 64 GB RAM and used two Nvidia 2080Ti GPU.

RL-based Motion Planning Training The RL-based motion planning model is trained using the CARLA simulator (Dosovitskiy et al. 2017) as an RL gym environment. The environment setup is vectorized, and the model trains by running the RL training simulation on different town maps in parallel. The training process consists of two phases: (a) the rollout phase and (b) the training phase. In the rollout phase, the AV keeps interacting with the environment for several episodes, trying to navigate around the environment and reach its destination. At each step, the AV collects the tuple $(\mathbf{O}, \mathbf{a}, r, \mathbf{O}', d)$ and stores it in its replay buffer. Here, \mathbf{O} and \mathbf{O}' are observations of timestep t and next timestep t' , \mathbf{a} is action, r is reward and d is done/termination condition. The AV mainly collects experience from various scenarios offered by different town maps. The AV remains in this phase until it has accumulated sufficient experience in the replay buffer for moving into the training phase. The training phase mainly involves the AV updating its policy network π_θ based on the experience it accumulated in the rollout phase. The policy π_θ is updated based on the loss function Equ.8, and the AV again reverts to the rollout phase. After a few rounds, the AV’s performance is evaluated, and several metrics are recorded to ensure that the AV is learning correctly and troubleshooting issues with proper tuning.

Performance Evaluation

Evaluation Environment Setup

To ensure a comprehensive evaluation of GP3Net, the performance benchmarking environment is setup based on the Leaderboard benchmarking suite described in the literature (Dosovitskiy et al. 2017) and NoCrash environment (Codevilla et al. 2019). These benchmarking suites offer diverse, complex scenarios across several towns, encompassing various weather conditions, numerical configurations, and behavior of vehicles and pedestrians. GP3Net’s performance is assessed and recorded after conducting several episodes involving different environment configurations to ensure that the model can generalize well in complex unseen scenarios.

Performance Evaluation Metrics

Specific performance evaluation metrics are employed to measure the performance of the predictive reinforcement learning-based motion planning framework GP3Net by the benchmarking suites aforementioned in the previous subsection. These metrics include the Success Rate (SR) and Driving Score (DS). The SR is calculated as the percentage of instances where the AV reaches its intended destination without experiencing any collision. This metric directly assesses the model’s ability to navigate and complete the designated

mission routes safely and successfully. The DS is a composite metric that combines the AV’s route completion success with penalties incurred due to collisions and traffic rule infractions. It is formulated as the product of the percentage of completed routes and a penalty factor.

Quantitative Benchmark Results

The GP3Net’s performance was evaluated in different towns and weather conditions for 100 episodes with five different seeds. Here Table 1 and Table 2 display the mean comparative results with other previously proposed solutions, some of which use a BEV perspective for motion planning similar to GP3Net. It can be observed in Table 1 that the predictive RL-based motion planner, GP3Net, outperforms DiGNet (Cai et al. 2021), its variations, and other works like CILRS (Codevilla et al. 2019) in terms of the success rate and the driving score in several towns. The mean improvement in the success rate and driving score is 3.85% and 8%, respectively. The standard deviation is 1.5%. GP3Net has a large improvement in Town 4 and Town 6 which consists of both mixed and highway environments, showing capability to handle well in higher speeds. This generalization to a wide range of scenarios can be attributed to the GP3Net’s understanding of how a non-stationary environment can evolve with the trajectory prediction module.

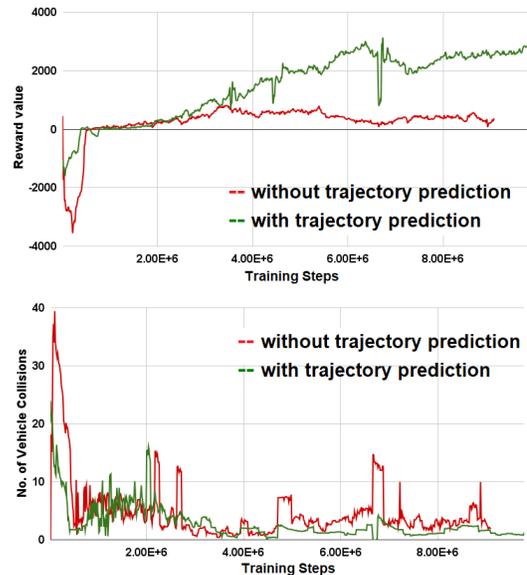


Figure 3: Plots of (a) reward obtained and (b) vehicle collisions with and without trajectory prediction module.

Another experiment was performed to test the resilience of GP3Net to different weather conditions, whose results are shown in Table 2. The test was conducted in Towns 1 and 2 under eight weather conditions mentioned in Table 2. According to Table 2, GP3Net is robust to rainy weather conditions and does a better job of navigating the environment successfully compared to another work, HIIL (Teng et al. 2023). The insights about the future trajectories provided by the proposed trajectory prediction module have positively

Models	Town 03 urban		Town 04 mixed		Town 05 urban		Town 06 highway		Town 01 urban		Town 02 urban	
	SR \uparrow	DS \uparrow	SR \uparrow	DS \uparrow	SR \uparrow	DS \uparrow	SR \uparrow	DS \uparrow	SR \uparrow	DS \uparrow	SR \uparrow	DS \uparrow
CILRS	43.2	0.56	16.2	0.28	38.2	0.52	35.2	0.48	42.3	-	24.7	-
MLP	62.3	0.72	60.5	0.70	60.0	0.69	72.2	0.79	-	-	-	-
GCN (U)	68.8	0.77	68.5	0.76	70.2	0.78	73.8	0.80	-	-	-	-
GCN (D)	74.2	0.80	64.0	0.72	71.5	0.79	74.8	0.81	-	-	-	-
GAT	76.0	0.82	71.0	0.78	81.5	0.87	78.2	0.83	-	-	-	-
DiGNet (CTL)	72.8	0.79	67.5	0.75	69.8	0.77	73.2	0.79	-	-	-	-
DiGNet (Full)	80.2	0.85	67.5	0.75	69.8	0.77	78.2	0.79	-	-	-	-
GP3Net (ours)	82.5	0.87	75.0	0.85	82.8	0.91	82.5	0.85	92.5	0.96	93.1	0.91

Table 1: A comparative result showing the mean success rate and the driving score of GP3Net and other SOTA works on different towns.

Weather Condition	T1 (ours)	T2 (ours)	T1 (HILL)	T2 (HILL)
Wetnoon	96	95.9	84	84
ClearSunset	97.3	92	88	88
ClearNoon	92	94	89	89
HardRainNoon	96	88	80	80
SoftRainSunset	93.3	94	86	86
WetSunset	98	94	85	85
WetCloudSunset	98	92	84	84
SoftRainNoon	88	91.4	88	88

Table 2: Success-rate Benchmarking results on T1: Town 1 and T2: Town 2 in different weather conditions.

impacted the RL-based motion planner’s decision-making during training and evaluation time, making it safe and cautious of the non-stationary environment. The following section discusses the effectiveness of our graph-based prediction module.

Qualitative Analysis

During training experiments, we plotted quantities such as total reward and the number of vehicle collisions for monitoring the training process for two cases: 1) With the trajectory prediction module and 2) without the trajectory prediction module. Fig.3 shows a significant difference in the training plots. After the 5 M steps, both the models have learned correctly not to deviate from the route or to stop when the lights turn red. However, the models were yet to learn to avoid collisions with other vehicles. The model with a trajectory prediction module showed a better capability to avoid collisions as the training progressed compared to the one which did not use a trajectory prediction module.

Ablation Study: In addition to the leaderboard suite, GP3Net’s performance was benchmarked on the NoCrash suite for analyzing the framework’s behavior in dense environments. Several metrics are utilized, such as end reach %, success rate %, penalty factor, crash rates, vehicle halts, and traffic lights passed were recorded to perform this analysis. The quantitative evaluation was done in Towns 1 and 2 in training and testing weather conditions which are dis-

played in Table 3. It can be observed that the success rate of GP3Net in Town 2, though competitive, is not as high as in Town 1 due to a much denser population of vehicles and walkers since Town 2 is smaller in size compared to Town 1. It is to be noted that the NoCrash suite has twice the walker population than that in the Leaderboard suite.

Performance Metric	T1 train	T1 new	T2 train	T2 new
End Reach (%)	100	100	96	90
Succ. Rate (%)	96	98	94	80
Driving Score	0.976	0.982	0.976	0.918
Penalty Factor	0.976	0.982	0.992	0.941
Score Route	1	1	0.984	0.977
Layout crash	0	0	0	0.047
Walker crash	0.039	0	0	0.151
Vehicle crash	0.018	0.039	0.083	0.167
Vehicle halt	0	0	0.313	1.556
Lights met	4.4	4.4	3.7	3.7
Lights passed	4.38	4.36	3.7	3.7

Table 3: Ablation study on CARLA NoCrash benchmarking suite. T1: Town 1; T2: Town 2.

Conclusion

This paper discusses the challenges faced by rule-based and imitation learning methods that rely on expert demonstrations. It highlights the limited ability of these methods to recover from domain shifts and near-failure scenarios in non-stationary environments. To mitigate this problem, this paper presents a deep Graph-based Prediction and Planning Network (GP3Net) framework that encodes spatio-temporal dynamic interactions of different traffic participants using a graphical model to infer their future behaviors and combines the advantage of exploration in reinforcement learning. GP3Net improves AV performance compared to the previous graph-encoded imitation learning-based policy in CARLA benchmarking driving scenarios for urban and highway environments. The qualitative study also clearly indicates the importance of the proposed prediction module for safe motion planning in dynamic driving scenarios. Fu-

ture work will look at other methods to model interactions with the dynamics of traffic participants involved and how they can influence the planning capabilities of AV.

References

- Aksjonov, A.; and Kyrki, V. 2021. Rule-Based Decision-Making System for Autonomous Vehicles at Intersections with Mixed Traffic Environment. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 660–666.
- Alizadeh, A.; Moghadam, M.; Bicer, Y.; Ure, N. K.; Yavas, U.; and Kurtulus, C. 2019. Automated Lane Change Decision Making using Deep Reinforcement Learning in Dynamic and Uncertain Highway Environment. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 1399–1404.
- Bansal, M.; Krizhevsky, A.; and Ogale, A. 2018. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*.
- Bazzi, A.; Berthet, A. O.; Campolo, C.; Masini, B. M.; Molinaro, A.; and Zanella, A. 2021. On the Design of Sidelink for Cellular V2X: A Literature Review and Outlook for Future. *IEEE Access*, 9: 97953–97980.
- Cai, P.; Wang, H.; Sun, Y.; and Liu, M. 2021. DiGNet: Learning Scalable Self-Driving Policies for Generic Traffic Scenarios with Graph Neural Networks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8979–8984.
- Cai, P.; Wang, S.; Sun, Y.; and Liu, M. 2020. Probabilistic End-to-End Vehicle Navigation in Complex Dynamic Environments With Multimodal Sensor Fusion. *IEEE Robotics and Automation Letters*, 5(3): 4218–4224.
- Chen, J.; Li, S. E.; and Tomizuka, M. 2021. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(6): 5068–5078.
- Chen, J.; Yuan, B.; and Tomizuka, M. 2019. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2765–2771. IEEE.
- Chen, L.; Hu, X.; Tang, B.; and Cheng, Y. 2020. Conditional DQN-based motion planning with fuzzy logic for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(4): 2966–2977.
- Choi, E.-H. 2010. Crash factors in intersection-related crashes: An on-scene perspective. Technical report.
- Chou, P.-W.; Maturana, D.; and Scherer, S. 2017. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, 834–843. PMLR.
- Codevilla, F.; Müller, M.; López, A.; Koltun, V.; and Dosovitskiy, A. 2018. End-to-End Driving Via Conditional Imitation Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 4693–4700.
- Codevilla, F.; Santana, E.; López, A. M.; and Gaidon, A. 2019. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9329–9338.
- Cui, J.; Qiu, H.; Chen, D.; Stone, P.; and Zhu, Y. 2022. Coopernaut: End-to-End Driving With Cooperative Perception for Networked Vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 17252–17262.
- Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.
- Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1861–1870. PMLR.
- Huang, X.; Hong, S.; Hofmann, A.; and Williams, B. C. 2019. Online risk-bounded motion planning for autonomous vehicles in dynamic environments. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 214–222.
- Ivanovic, B.; and Pavone, M. 2019. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2375–2384.
- Li, M.; and Zhu, Z. 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4189–4196.
- Ma, Y.; Zhu, X.; Zhang, S.; Yang, R.; Wang, W.; and Manocha, D. 2019. TrafficPredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 6120–6127.
- Naveed, K. B.; Qiao, Z.; and Dolan, J. M. 2021. Trajectory Planning for Autonomous Vehicles Using Hierarchical Reinforcement Learning. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 601–606.
- Petrović, D.; Mijailović, R.; and Pešić, D. 2020. Traffic accidents with autonomous vehicles: type of collisions, manoeuvres and errors of conventional vehicles’ drivers. *Transportation research procedia*, 45: 161–168.
- Rhinehart, N.; McAllister, R.; and Levine, S. 2020. Deep Imitative Models for Flexible Inference, Planning, and Control. In *International Conference on Learning Representations*.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Shi, L.; Wang, L.; Long, C.; Zhou, S.; Zheng, F.; Zheng, N.; and Hua, G. 2022. Social interpretable tree for pedestrian trajectory prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2235–2243.
- Tang, X.; Huang, B.; Liu, T.; and Lin, X. 2022. Highway Decision-Making and Motion Planning for Autonomous

Driving via Soft Actor-Critic. *IEEE Transactions on Vehicular Technology*, 71(5): 4706–4717.

Teng, S.; Chen, L.; Ai, Y.; Zhou, Y.; Xuanyuan, Z.; and Hu, X. 2023. Hierarchical Interpretable Imitation Learning for End-to-End Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 8(1): 673–683.

Xu, Y.; Yang, J.; and Du, S. 2020. CF-LSTM: Cascaded feature-based long short-term networks for predicting pedestrian trajectory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 12541–12548.

Ye, F.; Cheng, X.; Wang, P.; Chan, C.-Y.; and Zhang, J. 2020. Automated lane change strategy using proximal policy optimization-based deep reinforcement learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 1746–1752. IEEE.

Supplementary Materials for GP3Net

The supplementary materials consists of 1) More qualitative analysis 2) More quantitative results for different towns, 3) Architectural details of the neural network models used in GP3Net, 4) Hyper-parameter settings for GP3Net and 5) video recording of trained GP3Net based AV in different towns in different weathers.

Driving Videos by GP3Net

There are six videos included in the videos folder. The videos show many scenarios in different towns (small cities, big cities, highways, and mixed towns) and weather conditions (rainy and clear). There are urban intersections, junctions, highway lane merging, and diversions. The GP3Net performed better in all scenarios.

Qualitative Analysis

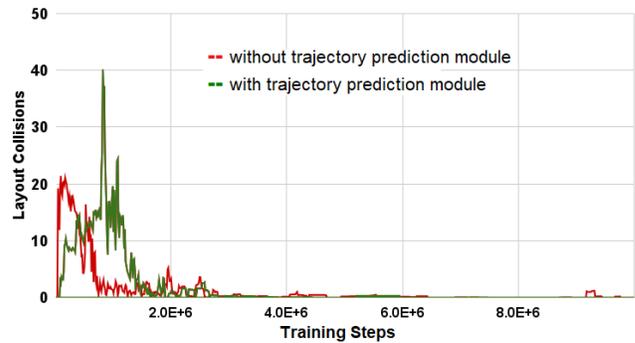


Figure 4: Plot of the layout collisions with and without trajectory prediction module.

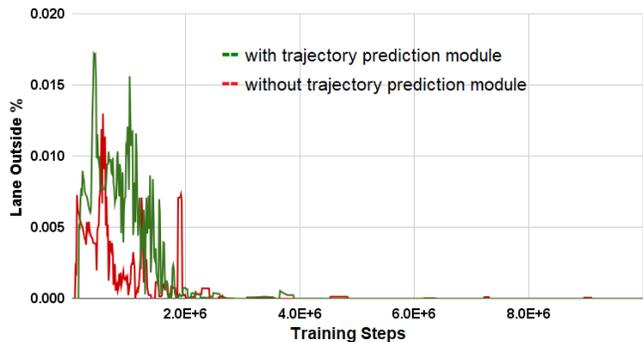


Figure 5: Plot of the percentage of cases going outside lane with and without trajectory prediction module.

This section shows some plots about the importance of the prediction module in motion planning for AVs. A PPO algorithm is trained without the prediction module to check its performance. The following plots show the performance in different metrics defined by CARLA with and without the trajectory prediction module.

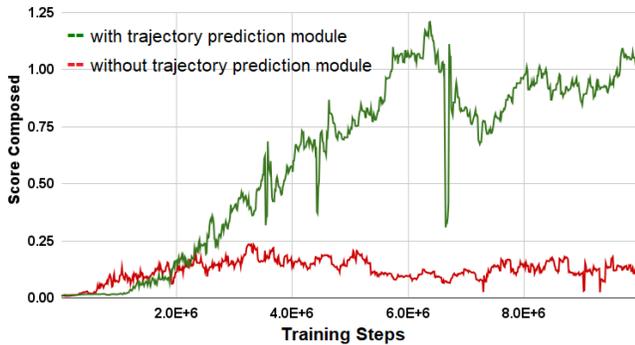


Figure 6: Plot of the score composed with and without trajectory prediction module.

In Fig.4 and Fig. 5, it can be seen that up to 2M steps, the GP3Net model without trajectory prediction performed better. However, after 2M training steps, the GP3Net with trajectory prediction module started performing well. In Fig.6, the advantage of the trajectory prediction module is visible. After 2M training steps, there is improvement in the metric score composed. This metric combines infractions and penalty factors and provides the final score for driving.

Detailed Quantitative Results

The detailed quantitative results for the CARLA Leaderboard benchmarking scenarios are given in Table.4.

Trajectory Prediction Architecture Details

The trajectory prediction module uses a deep generative trajectory prediction architecture which has Conditional Variational Auto Encoder (CVAE) to generate a distribution of potential trajectories. CVAE, a latent variable model, rely on Gaussian Mixture Models (GMM) to produce multimodal distribution outputs. The encoder and decoder of the CVAE consists of Long Short Term Memory (LSTM) networks for modeling temporal data such as trajectories and system evolution. After representing the nodes and edges as a vector, they are encoded using a Node Encoder and Edge Encoder respectively, whose hidden dimensions are shown in Table 5. The encoded nodes and edges are sent to an attention module and low-pass filter M . The probability density function $p(z|\mathbf{x})$ is captured by a Fully Connected Layer. Here, z is the latent variable and \mathbf{x} is the input to the model. After sampling z from $p(z|\mathbf{x})$, it is passed through the decoder LSTM network which outputs the parameters for the GMM, from which multimodal trajectories are sampled. Table 5 shows the Trajectory Prediction Architecture details.

State Encoder Architecture Details

The State Encoder consists of Convolutional Neural Network (CNN) for encoding the BEV (past and generated future) masks and a Multi-Layered Perceptron to encode the odometry values of AV. After concatenating the outputs from these two networks, it is further processed using a Fully-Connected Layer giving the final state encoding. The details of the State Encoder architecture is displayed in Table 6.

State Encoder Layers	IN Channels/ Dims	OUT Channels/ Dims	Filter Size
Conv 1 (BEV)	21	8	5
Conv 2 (BEV)	8	16	5
Conv 3 (BEV)	16	32	5
Conv 4 (BEV)	32	64	3
Conv 5 (BEV)	64	128	3
Conv 6 (BEV)	128	256	3
Fully Connected (Odometry)	256	256	-
Fully Connected (Odometry)	256	256	-
Fully Connected	256	512	-
Fully Connected	512	256	-

Table 6: State Encoder Architecture

PPO Policy Network Architecture Details and Hyperparameters

The policy network π like in many RL works is a Multi-Layered Perceptron (MLP). This policy network π gives the values of α and β parameters of Beta distribution of throttle and steering, resulting in two α s and two β s. The details for policy network architecture and hyperparameters used for PPO training is shown in Table 7.

Policy Network Layers	IN dims	OUT dims
Fully Connected	256	256
Fully Connected (α)	256	2
Fully Connected (β)	256	2
PPO Hyperparameters		Values
Clip ratio ϵ		0.2
Epochs		30
Gamma γ		0.99
Target KL		0.01
GAE λ		0.97
Learning Rate		0.00003

Table 7: Policy Network Architecture and Hyperparameter Details

Data collection settings for trajectory prediction module

The training data was collected using the CARLA simulator’s Autopilot/Roaming Agent in Town 1. For several episodes, this autopilot vehicle navigates from source to destination, recording the trajectories of all vehicles at each time step t . The trajectories are represented by a sequence of local rectangular coordinates $\mathbf{x}_t = (x_t, y_t)$ of the agents at time step t , as observed by the roaming agent. The dataset contains the timestamp of the recorded trajectory data, the index of the agent, and the trajectory coordinates. The data for vehicles and pedestrians are collected every 4-time steps (0.4 seconds) and stored in separate datasets, as we train two models for the former and the latter.

Performance Metric	Town01 Urban	Town02 Urban	Town03 Urban	Town04 Mixed	Town05 Urban	Town06 Highway
End Reach (%)	100	95.43	96.25	93.75	97.86	95.0
Success Rate (%)	92.5	93.14	82.5	75.0	82.8	82.5
Driving Score	0.96	0.91	0.87	0.85	0.91	0.91
Penalty Factor	0.9588	0.9840	0.9062	0.8721	0.9223	0.914
Score Route	1.0	0.9994	0.9866	0.9794	0.9916	0.9832
Layout Crash	0	0	0	0.0115	0	0
Walker Crash	0.0922	0	0.2141	0.0419	0.015	0.0104
Vehicle Crash	0	0.0508	0.0320	0.0579	0.1291	0.0865
Vehicle Halt	0	0.2509	0.0394	0.0439	0.0572	0.0296
Lights met	4.7	3.76	7.638	8.65	8.2357	6.025
Lights passed	4.675	3.7371	7.5375	8.5375	8.2214	5.975

Table 4: Detailed performance of GP3Net on Leaderboard Benchmarking scenarios in different towns

Network Architecture Components	Value
Batch Size	16
Learning Rate	0.001
Minimum Learning Rate	0.00001
Learning Decay Rate	0.9999
Edge Addition distance (vehicles)	45 meters
LSTM Encoder Edge	8
LSTM Node History Encoder Hidden Dimensions	32
LSTM Node Future Encoder Hidden Dimensions	32
FC Layer $p(z \mathbf{x})$ Dimensions	16
LSTM Decoder Hidden Dimensions	128
GMM Components	16
Past Data Length	8 steps (3.2 sec)
Future Prediction Horizon	7 steps (2.8 sec)

Table 5: Trajectory Prediction Module Architecture and Hyperparameters Details