AGen: Adaptable Generative Prediction Networks for Autonomous Driving

Wenwen Si, Tianhao Wei, and Changliu Liu

***Abstract*— In highly interactive driving scenarios, accurate prediction of other road participants is critical for safe and efficient navigation of autonomous cars. Prediction is challenging due to the difficulty in modeling various driving behavior, or learning such a model. The model should be interactive and reflect individual differences. Imitation learning methods, such as parameter sharing generative adversarial imitation learning (PS-GAIL), are able to learn interactive models. However, the learned models average out individual differences. When used to predict trajectories of individual vehicles, these models are biased. This paper introduces an adaptable generative prediction framework (AGen), which performs online adaptation of the offline learned models to recover individual differences for better prediction. In particular, we combine the recursive least square parameter adaptation algorithm (RLS-PAA) with the offline learned model from PS-GAIL. RLS-PAA has analytical solutions and is able to adapt the model for every single vehicle efficiently online. The proposed method is able to reduce the root mean squared prediction error in a $2.5\,\mathrm{s}$ time window by 60%, compared with PS-GAIL.**

I. Introduction

In order to achieve safe and high-quality decision making and motion planning, autonomous vehicles should be able to generate accurate probabilistic predictions for uncertain behavior of other road users. Prediction is challenging due to the difficulty in modeling various driving behavior, or learning such a model from data. The model should be *interactive* to consider interactions among road users, and *heterogeneous* to reflect individual differences.

Generative adversarial imitation learning (GAIL) [1], as a direct extension of generative adversarial network (GAN) [2], has been successfully applied to obtain vehicle policies in car-following and highway-driving [3]. A driving policy is a model that inputs the current world state and outputs the next action (*e.g.*, longitudinal acceleration and turning rate). Such a model is generative in the sense that we can simulate future vehicle trajectories using the learned driving policy to make predictions. The simulation process is called *roll-out*. In addition to the generative model (*e.g.*, the policy), GAIL has an adversarial component, called a critic, to compute the difference between the roll-out trajectories and true trajectories. During training, the algorithm iteratively updates the policy to minimize the difference, and updates the critic to maximize the difference. When the algorithm converges, the critic cannot distinguish between the true trajectory and the roll-out trajectory, which implies point-wise convergence of the policy (*i.e.*, the model). However, when evaluated in a multi-agent setting, the policies learned through single-agent imitation learning fail to make accurate predictions due to covariate shift [4]. Covariate shift is a situation when the distribution of scenarios for testing does not match the distribution of scenarios for training. Intuitively, single agent GAIL does not consider others' responses during interactions, hence results in errors when evaluated in multi-agent scenarios.

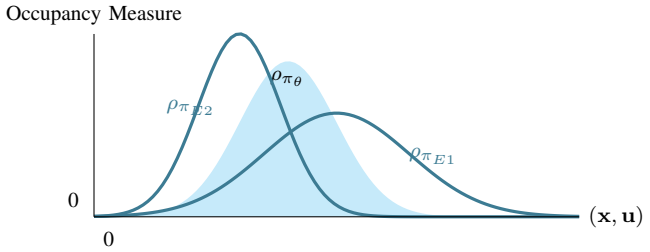


Fig. 1: The averaging effect of parameter sharing. When the policy network is parameterized by shared parameters, *e.g.*, in PS-GAIL, the distributions of occupancy measures of different drivers $\rho_{\pi_{E1}}$ and $\rho_{\pi_{E2}}$ is averaged to be ρ_{π_θ}. Our proposed method will learn to fit individual distributions $\rho_{\pi_{E1}}$ and $\rho_{\pi_{E2}}$ as they are. An occupancy measure captures the probability for a vehicle to take certain actions $\mathbf{u}$ in certain states $\mathbf{x}$.

Multi-agent GAIL (MA-GAIL) [5] is needed to build reliable models that consider multi-vehicle interaction. As it is computationally intractable to learn a specific model for every vehicle, parameter sharing GAIL (PS-GAIL) [3], as a special case of (MA-GAIL), is introduced. PS-GAIL assumes that all vehicles are homogenous. It learns an interactive model in multi-agent environments. PS-GAIL achieves a notable improvement on multi-agent trajectory prediction compared to single agent GAIL. However, homogeneity among vehicle behavior does not hold for real world driving. The model learned by PS-GAIL effectively averages out various driving behavior as shown in Fig. 1. It fails to capture individual differences and creates bias in the prediction. Moreover, as PS-GAIL learns a constant model, it also cannot capture time-varying behavior, *e.g.*, change of mind. These problems are not unique to models produced by GAIL families. All constant and homogeneous models suffer from these problems:

1) A constant model fails to capture the time varying behavior of vehicles;
2) A homogeneous model fails to consider individual differences among various vehicles (drivers).

To solve these problems, this paper introduces an online adaptation framework for heterogeneous prediction, which

*This project is supported by Holomatic.

W. Si, T. Wei, and C. Liu are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA (e-mail: wenwens, twei2, cliu6@andrew.cmu.edu).

leads to an adaptable generative prediction model (AGen). We combine the state-of-the-art multi-agent generative model learned by PS-GAIL with recursive least square parameter adaptation algorithms (RLS-PAA) [6] to recover individual differences for better prediction. An earlier work that applies RLS-PAA on learning-based human motion prediction is discussed in [7]. RLS-PAA has analytical solutions and is able to adapt the model for every vehicle efficiently online. The proposed method is able to reduce the root mean squared prediction error in a $2.5\,\mathrm{s}$ time window by 60%. Our code is publicly available at `https://github.com/intelligent-control-lab/AGen`.

The remainder of the paper is organized as follows. Section II formulates the prediction problem. Section III proposes our methodology. Section IV shows the performance of the proposed method and comparison with other methods. Section V discusses limitations and extensions of the proposed method. Section VI concludes the paper.

II. Problem Formulation

We call the vehicle making the prediction as *the ego vehicle*, the vehicle to be predicted as *the target vehicle*. For a target vehicle, denote its state space as $\mathcal{X}$, its observation space as $\mathcal{O}$, and its action space as $\mathcal{U} \subset \mathbb{R}^2$. The observation space includes the state of the target vehicle $\mathbf{x} \in \mathcal{X}$ (*e.g.*, location, velocity, and acceleration) as well as the world state (*e.g.*, lane position, state of other road participants).[1] The action space is for longitudinal acceleration and turning rate.

To make reliable predictions, a vehicle behavior model is needed. We choose to use a vehicle policy as our model. A vehicle policy $\boldsymbol{\pi} : \mathcal{O} \to \mathcal{U}^2$ maps an observation $\mathbf{o}(k) \in \mathcal{O}$ at time k to an action $\mathbf{u}(k+1) \in \mathcal{U}$.[3] Trajectory can be predicted by forward simulation using the policy. The simulator has a dynamic function $\mathbf{s} : \mathcal{U} \to \mathcal{O}$ that maps an action $\mathbf{u}(k+1) \in \mathcal{U}$ to the observation $\mathbf{o}(k+1) \in \mathcal{O}$ at time $k+1$. Through forward simulation at time k, we obtain a sequence of action and observation pairs in the future

$$\mathbf{o}(k) \to_{\boldsymbol{\pi}} \mathbf{u}(k+1) \to_{\mathbf{s}} \mathbf{o}(k+1) \to_{\boldsymbol{\pi}} \mathbf{u}(k+2) \cdots . \quad (1)$$

Since the observation $\mathbf{o}$ contains the state of the target vehicle $\mathbf{x}$, the future trajectory $\mathbf{x}(k+i)$ for $i \geq 0$ of the target vehicle can be predicted. This process is called roll-out. The vehicle policy $\boldsymbol{\pi}$ is essential for accurate predictions. This paper adopts imitation learning to identify $\boldsymbol{\pi}$. Refer to appendix for preliminaries on imitation learning.

[1]The observation space of the target vehicle may be only partially known to the ego vehicle. This paper assumes full observability from the ego vehicle. The influence of partial observability will be studied in the future.

[2]In stochastic scenarios, $\boldsymbol{\pi}$ maps an observation $\mathbf{o}$ to a distribution over the action space. As such a distribution is assumed to be Gaussian and we always consider the most likely action in the prediction, we simplify the notation to the non-stochastic version. The policy always map to the most likely action, which is the mean value in the Gaussian distribution.

[3]We shift the time indices for control input one step ahead compared to conventional dynamic system definition. The control input applied at time k is denoted $\mathbf{u}(k+1)$, since it affects the state and observation at time $k+1$.

Algorithm 1: AGen Algorithm

input : History trajectories and actions, pretrained policy network in (3)
output : Predicted trajectories and actions, heterogenous models in (2)
parameter : length of history N, forgetting factor λ, prediction horizon l
Initialization:

while *True* **do**
  $k = k + 1$;
  for *all vehicles* i **do**
    Obtain new observation $\mathbf{o}_i(k)$;
    Construct N-step regressor $\boldsymbol{\Phi}_i(k)$ in (11b);
    Compute prediction error $\boldsymbol{\epsilon}(k)$in (10b) given observed action $\mathbf{u}_i(k)$;
    Adapt the parameter $\mathbf{W}_i(k) \leftarrow$ `RLS-PAA` $(\boldsymbol{\Phi}_i(k-1), \boldsymbol{\epsilon}(k), \mathbf{W}_i(k-1))$ using (9) and (10);
  end
  Trajectory roll-out in simulation (1) ;
end

There are mainly two reasons why we do not use a fixed-horizon trajectory as direct output. First, a generative model can predict trajectories of arbitrary length. Second, the learned policy or the generative model can be directly used by the ego vehicle to plan and control its motion for autonomous driving.

In a multi-vehicle scenario, we would like to simultaneously predict several vehicles. Denote the parameters of different target vehicles using subscript $i \in \mathbb{N}$. Vehicles are heterogeneous. Instead of fitting an average policy model $\boldsymbol{\pi}$ for all vehicles, we need to specify a policy model $\boldsymbol{\pi}_i$ for every target vehicle to account for individual differences.

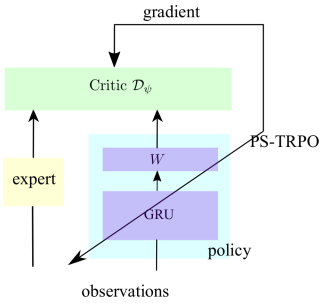
To obtain a prediction model that is both interactive and heterogeneous, and to explicitly incorporate historical information, we will identify a function $\mathbf{f} : \mathcal{O} \times \mathcal{H} \times \mathbb{N} \to \mathcal{U}$, which outputs the action $\mathbf{u}_i(k+1) \in \mathcal{U}$ of individual vehicle i given current observation $\mathbf{o}_i(k) \in \mathcal{O}$ and truncated history $\boldsymbol{\Phi}_i(k) \in \mathcal{H}$,

$$\mathbf{u}_i(k+1) = \mathbf{f}(\mathbf{o}_i(k), \boldsymbol{\Phi}_i(k), i). \quad (2)$$

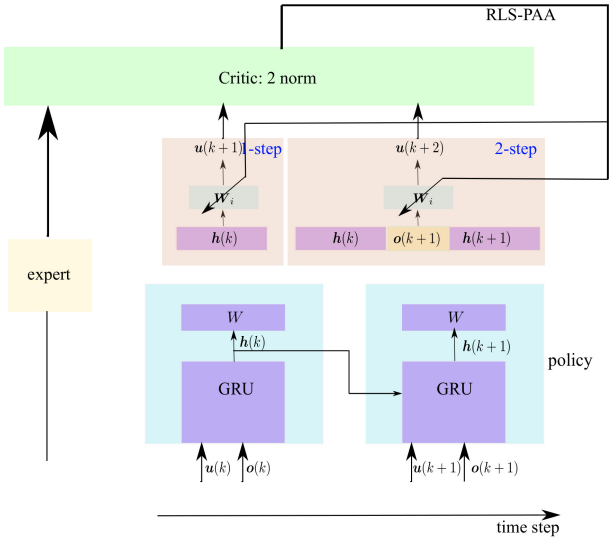
The function $\mathbf{f}$ is our behavior model which varies for different vehicles. For vehicle i, $\mathbf{f}(\cdot, \cdot, i)$ is different from the policy $\boldsymbol{\pi}_i$, in that $\mathbf{f}$ directly involves the history $\boldsymbol{\Phi}_i(k)$. The history $\boldsymbol{\Phi}_i(k)$ corresponds to the hidden state in our policy network and is explicitly used in the online adaptation, to be discussed in detail in section III.

III. Methodology

Heterogeneity among drivers needs to be explicitly accounted to improve prediction accuracy in real world scenarios. As mentioned earlier, it is intractable to fit a policy network for every individual vehicle. To make heterogeneous prediction scalable, we combine offline model learning with



(a) Offline training using PS-GAIL. Critic computes the difference between the expert trajectory and the roll-out trajectory from the policy network. PS-GAIL iteratively updates the policy to minimize the difference and the critic to maximize the difference.



(b) Online adaptation using RLS-PAA. The critic computes the 2-norm difference between the the expert trajectory and the roll-out trajectory from the policy network. RLS-PAA updates the policy network (only the last layer) to minimize the difference. We can either do 1-step or 2-step adaptation.

Fig. 2: Illustration of offline training of the policy model using PS-GAIL and online adaptation of the policy model using RLS-PAA. The policy model contains GRU modules and a fully connected last layer parameterized by $\mathbf{W}$.

online model adaptation. Offline model learning extracts features for average driving behavior. Online model adaptation can perturb the average model to fit the behavior of a specific driver at a specific time. In particular, we take the offline pretrained policy network of PS-GAIL as the feature extractor for averaged driving behavior, while adapting individual vehicle behavior using RLS-PAA online. The online adaptation algorithm is shown in algorithm 1.

A. Offline Feature Extraction using PS-GAIL

Offline training using PS-GAIL is illustrated in Fig. 2a. It consists of a basic GAN structure. A discriminator $\mathcal{D}_\psi$ acts as the critic that surrogates a reward function in the environment. The policy network $\bar{\mathbf{f}} : \mathbb{R}^m \mapsto \mathbb{R}^2$ takes in observations of the environment[4] and synthesizes action sequences that emulate the behavior of expert distribution. We model the policy network using a recurrent neural network (RNN), which further takes in the hidden vector $\mathbf{h} \in \mathbb{R}^n$ and previous input $\mathbf{u} \in \mathbb{R}^2$, *i.e.*, [5]

$$\mathbf{u}_i(k+1) = \bar{\mathbf{f}}(\mathbf{o}_i(k), \mathbf{h}_i(k-1), \mathbf{u}_i(k)), \tag{3a}$$

$$= \mathbf{W} \underbrace{\mathbf{g}(\mathbf{o}_i(k), \mathbf{h}_i(k-1), \mathbf{u}_i(k))}_{\mathbf{h}_i(k)}, \tag{3b}$$

where $\mathbf{g} : \mathbb{R}^m \mapsto \mathbb{R}^n$ can be regarded as a feature extractor that computes the hidden vector. $\mathbf{W}$ is the parameter of the last layer that maps the hidden vector to the output. The model $\bar{\mathbf{f}}$ is equivalent to (2) except that it remains the same for all vehicles (due to parameter sharing).

At each iteration of the algorithm, the policy with shared parameters is used by each agent to generate trajectories. Rewards are then assigned to each state-action pair in these trajectories by the critic. Subsequently observed trajectories are used to perform a TRPO [8] update of the policy, and an Adam [9] update of the critic.

Due to parameter sharing, $\bar{\mathbf{f}}$ is an average model. Nonetheless, the hidden layers of $\bar{\mathbf{f}}$ can extract useful features for different driving behavior.

B. Online Adaptation Using Recursive Least Square

We use RLS-PAA to adapt parameters for individual vehicles. RLS-PAA is efficient for identification of time-varying systems. Specifically, the system to be adapted online is the function from the hidden layer $\mathbf{h}_i(k)$ to the output layer $\mathbf{u}_i(k+1)$ for every i. It is a linear system in the open loop as shown in (3). However, due to nonlinearity of function $\mathbf{g}$, the closed loop system is nonlinear. In the following discussion, we first discuss how RLS-PAA can be applied to a linear system as shown in Fig. 3. In sections III-C and III-D, we provide two approaches to handle the nonlinearity.

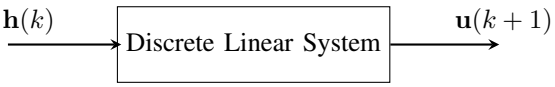


Fig. 3: Benchmark problem for online adaptation using RLS-PAA.

1) Benchmark problem for RLS-PAA: Suppose the transfer function of the discrete linear system shown in Fig. 3 can be parameterized as

$$\mathbf{u}(k+1) = \frac{\sum_{j=0}^{q} b_j z^{-j}}{1 + \sum_{j=1}^{p} a_j z^{-j}} \mathbf{h}(k), \tag{4}$$

where a_j and b_j are unknown parameters for all j, which may be time varying. And z^{-1} represents one step lag, *i.e.*, $\mathbf{u}(k) = z^{-1}\mathbf{u}(k+1)$. To identify the unknown parameters, (4) is transformed to the following representation,

$$\mathbf{u}(k+1) = \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{\phi}(k), \tag{5}$$

[4]The dimension of the observation space is chosen to be $m = 66$ in the experiment. Refer to [3] for details on the selection of the features.

[5]The dimension of the hidden space is chosen to be $n = 64$ in the experiment.

where $\boldsymbol{\theta}$ is the parameter vector and $\boldsymbol{\phi}$ is the regressor vector. They are defined as

$$\boldsymbol{\theta} = [a_1, a_2, \ldots, a_p, b_0, b_1, \ldots, b_q]^\mathsf{T}, \tag{6a}$$

$$\boldsymbol{\phi}(k) = [-\mathbf{u}(k), \ldots, -\mathbf{u}(k+1-p), \mathbf{h}(k), \ldots, \mathbf{h}(k-q)]^\mathsf{T}. \tag{6b}$$

Equation (5) is the benchmark problem of RLS-PAA, which can be regarded as a linearization of (3).

2) Solving the benchmark problem: At time k, the following least square problem is formulated to estimate $\boldsymbol{\theta}$,

$$\hat{\boldsymbol{\theta}}(k) := \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{k} \lambda^{k-j} \|\mathbf{u}(j) - \boldsymbol{\theta}^\mathsf{T} \boldsymbol{\phi}(j-1)\|^2, \tag{7}$$

where $\lambda \in (0, 1]$ is a forgetting factor. Analytically, the estimate $\hat{\boldsymbol{\theta}}(k)$ can be computed as,

$$\hat{\boldsymbol{\theta}}(k) = \left[\sum_{j=1}^{k} \lambda^{k-j} \boldsymbol{\phi}(j-1)\boldsymbol{\phi}^\mathsf{T}(j-1)\right]^{-1} \sum_{j=1}^{k} \lambda^{k-j} \boldsymbol{\phi}(j-1)\mathbf{u}(j). \tag{8}$$

To get rid of the matrix inversion, $\hat{\boldsymbol{\theta}}(k)$ can be obtained recursively [6],

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \mathbf{F}(k)\boldsymbol{\phi}(k)\boldsymbol{\epsilon}(k+1), \tag{9}$$

where $\mathbf{F}(k)$ is the learning gain and $\boldsymbol{\epsilon}(k+1)$ is the prediction error with

$$\mathbf{F}(k+1) = \mathbf{F}(k) - \frac{\lambda \mathbf{F}(k)\boldsymbol{\phi}(k)\boldsymbol{\phi}(k)^\mathsf{T}\mathbf{F}(k)}{1 + \lambda \boldsymbol{\phi}(k)^\mathsf{T}\mathbf{F}(k)\boldsymbol{\phi}(k)}, \tag{10a}$$

$$\boldsymbol{\epsilon}(k+1) = \mathbf{u}(k+1) - \hat{\boldsymbol{\theta}}^\mathsf{T}(k)\boldsymbol{\phi}(k). \tag{10b}$$

Compared with other online adaptation algorithm, such as stochastic gradient descent [10], RLS-PAA is able to get an analytical solution that locally fits the data. This local fitting is beneficial for a predictor to emulate the sudden change of drivers' mind or driving environment.

C. AGen: Adaptable Generative Model

AGen applies RLS-PAA-based online adaptation to an offline learned model to emulate individual differences. For online adaptation, AGen takes in the hidden vector extracted by the policy network of PS-GAIL and outputs the actions and the roll-out trajectory, as shown in algorithm 1. It adapts the mapping from the hidden vector $\mathbf{h}$ (*i.e.*, offline extracted features) to the predicted action $\mathbf{u}$. As discussed in section III-B, the closed loop system between the hidden vector $\mathbf{h}$ and $\mathbf{u}$ is nonlinear. To apply RLS-PAA, we simply linearize (2) and (3) and obtain

$$\mathbf{u}_i(k+1) = \mathbf{W}_i(k)\boldsymbol{\Phi}_i(k), \tag{11a}$$

$$\boldsymbol{\Phi}_i(k) = [\mathbf{o}_i(k), \mathbf{h}_i(k), \mathbf{h}_i(k-1), \ldots, \mathbf{h}_i(k-N+1)], \tag{11b}$$

where $\boldsymbol{\Phi}_i(k)$ is the concatenate of current observations $\mathbf{o}_i(k)$ and the hidden vectors in previous N steps. The hidden vector $\mathbf{h}_i(k)$ is defined in (3). The matrix $\mathbf{W}_i$ corresponds to $\boldsymbol{\theta}$ in (5), which is to be identified online to account for the unique behavior of vehicle i. The matrix $\mathbf{W}_i$ can be regarded as the last layer of the policy network $\bar{\mathbf{f}}$, though its column dimension may not match $\mathbf{W}$ in (3).

As a special case, we consider $\boldsymbol{\Phi}_i(k) = \mathbf{h}_i(k)$. It is called 1-step adaptation since $N = 1$. The observation $\mathbf{o}_i(k)$ is not considered. As a result, $\mathbf{W}_i$ in (11) corresponds to $\mathbf{W}$ in (3). The offline learned policy $\bar{\mathbf{f}}$ in (3) is an average of individual policies in 1-step AGen (11) as shown in Fig. 1, *i.e.*,

$$\mathbf{W}\mathbf{g} = \bar{\mathbf{f}} \equiv \overline{\mathbf{W}_i}\mathbf{g}, \tag{12}$$

where $\overline{\mathbf{W}_i}$ denotes the average over all $\mathbf{W}_i$. Due to adaptation of the last layer's weights, the prediction in 1-step AGen creates a bias through $\mathbf{W}_i$ to account for heterogeneity of individual vehicles. Hence the prediction through 1-step AGen is more accurate than the prediction directly obtained from the offline model, which will be shown in section IV. Though oversimplified, 1-step AGen is computationally efficient, while still be able to capture individual differences.

D. RLS-PAA on Feedback Recurrent Neural Network

In the general formulation of the feature vector $\boldsymbol{\Phi}$ in (11b), we include observation $\mathbf{o}$. In the open loop, the observation $\mathbf{o}$ only affects the output $\mathbf{u}$ through the hidden vector $\mathbf{h}$. However, to linearize the closed loop system and apply multi-step RLS-PAA, we need to include both $\mathbf{o}$ and h, which will be explained in detail below.

For multi-step RLS-PAA, due to the feedback structure of RNN, the output action is getting back into the RNN as part of the new observation. In particular, the structure of the policy network that uses Gated Recurrent Units (GRUs) is shown in Fig. 4.

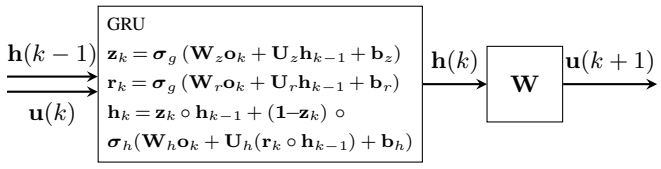


Fig. 4: Flow chart of the system with a feedback GRU-based RNN as input features extractor.

Due to the complex structure of the GRU cell, the feedback action $\mathbf{u}(k+1)$ and the hidden vector $\mathbf{h}(k)$ are coupled in a twisted nonlinear function, with no explicit decomposition. Hence, it is difficult to put the system in the form of the benchmark RLS-PAA problem (5). In order to make better use of truncated history with longer window, we introduce a *blocking* technique to approximately tackle this dilemma. Under the blocking mechanism, the history of $\mathbf{u}(j)$ for $j \leq k$ is lumped to the observation $\mathbf{o}(k)$. Hence the regressor $\boldsymbol{\phi}(k)$ in (6b) is approximated by the regressor $\boldsymbol{\Phi}(k)$ in (11b).

For the feedback action, we use the action in the ground truth trajectory to block the backward arrow when updating the adaptive parameters. The procedure of the general multi-step AGen is shown in Fig. 2b. The blocking technique for multi-step AGen is inspired by [11], which applies RLS-PAA on RNNs without output-to-input feedbacks and results in promising training performance. It is worth noting that 1-step AGen discussed earlier corresponds to an open-loop

approximation of (3), while multi-step AGen concerns with a closed-loop approximation.

IV. Experimental Results

A. Experimental Setup

AGen is validated on US 101 human driving data from the Next Generation SIMulation (NGSIM) dataset [12]. It is a widely used benchmark dataset for autonomous driving problem. The vehicle trajectories are captured by cameras mounted on top of surrounding buildings.

The feature extractor $\mathbf{g}$ in (3) is obtained from PS-GAIL. It is an RNN consists of 64 GRUs. The observation is passed directly into the RNN without any initial reduction in dimensionality. The critic is implemented as a feed-forward neural network consisting of $(128, 128, 64)$ ReLU units implemented as a Wasserstein GAN with gradient penalty (WGAN-GP) with a gradient penalty of 2.

For online adaptation, we compare the performance of 1-step AGen discussed in section III-C and 2-step AGen discussed in section III-D. The prediction horizon is $l = 25$. The sampling time is $0.1\,\mathrm{s}$. Our algorithm predicts a trajectory of 25 points in a $2.5\,\mathrm{s}$ window. For the 25 steps in the prediction, only the first observation is obtained from the ground truth data, while consequent inputs to the feature extractor $\mathbf{g}$ are all generated from the simulator during trajectory roll-out based on the predicted actions by AGen. The simulator simply considers kinematics of the vehicles. The forgetting factor is chosen to be $\lambda = 0.99$.

B. Evaluation Metrics

The predicted roll-out trajectories are compared to ground truth trajectories. We evaluate the performance of the prediction using root mean squared error (RMSE) of the predicted variable v,

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(v_t^{(i)} - \hat{v}_t^{(i)}\right)^2} \tag{13}$$

where $v_t^{(i)}$ is the true value in the ith trajectory at time t and $\hat{v}_t^{(i)}$is the simulated roll-out value for the ith trajectory at time t. m is the total number of trajectories. We extract the RMSE in predictions of global position, lane offset, and speed over time horizons up to $2.5\,\mathrm{s}$.

We also evaluate the longitude and lateral RMSE separately, with statistics including standard deviation and top 5% and top 10% error.

C. Prediction Results

1) Computation time: All experiments are performed with single thread on 3.1 GHz Intel Core i7 Processor. We perform both single agent tests and multi-agent tests. In single agent tests, only one vehicle is predicted while other vehicles' trajectories are assumed to be known. In multi-agent tests, the trajectories of all vehicles are predicted simultaneously. The average computation time for AGen in one time step is $0.083\,\mathrm{s}$ in single agent tests, and $0.79\,\mathrm{s}$ in multi-agent tests with 22 agents. The computation time grows nonlinearly with respect to the number of agents, due to the coupling effects in multi-agent scenarios.

TABLE I: Average Position RMSE on a $2.5\,\mathrm{s}$ Time Span.

Average	1 Agent		22 Agents	
RMSE (m)	Longitude	Lateral	Longitude	Lateral
PS-GAIL	1.13	0.38	1.33	0.44
1-step AGen	0.42	0.19	0.43	0.18
2-step AGen	0.56	0.26	0.56	0.26

TABLE II: Top Position RMSE on a $2.5\,\mathrm{s}$ Time Span.

Average	1 Agent		22 Agents	
RMSE (m)	top 5%	top 10%	top 5%	top 10%
PS-GAIL	5.03	4.29	5.68	4.96
1-step AGen	2.36	1.92	2.35	1.91

2) RMSE: The average position RMSE over a $2.5\,\mathrm{s}$ time span is shown in table I. The top error statistics are shown in table II. The entry with "PS-GAIL" refers to the prediction results by directly using the offline trained model from PS-GAIL without any online adaptation. As shown in the tables, AGen significantly reduces the prediction error. The predicted trajectories for the 22-agent scenario is shown in Fig. 5. Fig. 6 shows the growth of RMSE with respect to time.

V. Discussion

The major limitation of the proposed method is that it requires omniscient perspective. However, in real world driving, the ego vehicle j cannot fully obtain the observation $\mathbf{o}_i$ of the target vehicle i. This partial observability may cause instability in online adaptation. Methods to deal with partial observability will be explored in the future.

Another limitation of the proposed method is that RLS-PAA takes time to converge to true values. However, in real world driving situations, the interactions between two vehicles may not last long, which may finish before the con-

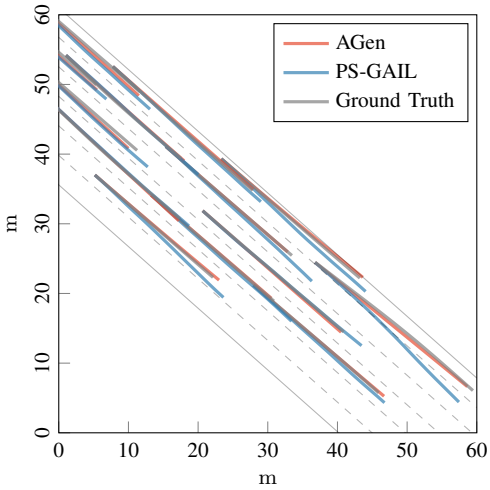


Fig. 5: Predicted $2\,\mathrm{s}$ trajectories for 22 agents after $3\,\mathrm{s}$ adaptions.

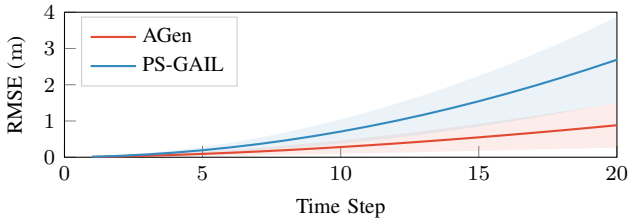


Fig. 6: Average position RMSE over time in the 22-agent scenario in Fig. 5. The shaded areas indicate the standard deviation of the prediction error.

vergence of the algorithm. Methods to analyze and improve the convergence of the algorithm will be studied in the future.

Another approach to account for heterogeneity in prediction is to add more prior knowledge through conditional probability, *e.g.*, infoGAN [13] and infoGAIL [14]. These methods are able to distinguish individual differences during offline training. Using these models, we will be able to maximize the information for online adaptation, which could provide better initial approximation than homogeneous models. Nonetheless, infoGAIL still takes time average. We can leverage the advantages of infoGAIL and online adaptation by combining the two, in order to account for time-varying behavior and to make the online adaptation algorithms converge faster.

VI. Conclusion

This paper proposed an adaptive generative method (AGen) for vehicle trajectory prediction. AGen combined RLS-PAA-based online adaptation with pretrained policy network from PS-GAIL to account for individual differences and time-varying behavior. The design considerations for both single-step and multi-step adaptation systems were elaborated. Experiments on real world driving dataset verified the effectiveness of the method.

References

[1] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[3] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *International Conference on Intelligent Robots and Systems*, IEEE, 2018, pp. 1534–1539.

[4] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[5] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 7461–7472.

[6] V Panuska, "An adaptive recursive-least-squares identification algorithm," in *Symposium on Adaptive Processes (8th) Decision and Control*, IEEE, 1969, pp. 65–65.

[7] Y. Cheng, W. Zhao, C. Liu, and M. Tomizuka, "Human motion prediction using semi-adaptable neural networks," in *American Control Conference*, 2019, arXiv:1810.00781.

[8] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[10] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[11] Q. Song, X. Zhao, Z. Feng, and B. Song, "Recursive least squares algorithm with adaptive forgetting factor based on echo state network," in *World Congress on Intelligent Control and Automation*, IEEE, 2011, pp. 295–298.

[12] J. Colyar and J. Halkias, "US highway 101 dataset," *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[13] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016, pp. 2172–2180.

[14] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3812–3822.

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv:1701.07875*, 2017.

Appendix
Preliminaries on Imitation Learning

Imitation learning is to learn a policy $\boldsymbol{\pi}$ that imitates an expert policy $\boldsymbol{\pi}_E$ given demonstrations generated by the expert policy. A demonstration is a sequence of state-action pairs $(\mathbf{x}, \mathbf{u})$ that result from a policy interacting with the environment.

Generative adversarial imitation learning (GAIL) [1] formulates imitation learning as a problem to match the state-action occupancy distribution of the expert policy. For a policy π, its occupancy measure $\rho_\pi : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is defined as $\rho_\pi(\mathbf{x}, \mathbf{u}) = \mathbb{P}_{\boldsymbol{\pi}}(\mathbf{u} \mid \mathbf{x}) \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_{\boldsymbol{\pi}}(\mathbf{x}(k) = \mathbf{x})$ where $\mathbb{P}_{\boldsymbol{\pi}}(\mathbf{u} \mid \mathbf{x})$ is the probability of taking action $\mathbf{u}$ at state $\mathbf{x}$ given the policy $\boldsymbol{\pi}$, $\mathbb{P}_{\boldsymbol{\pi}}(\mathbf{x}(k) = \mathbf{x})$ is the probability of landing in state $\mathbf{x}$ at time k under policy $\boldsymbol{\pi}$, γ is the discount factor. The occupancy measure can be interpreted as the distribution of state-action pairs that an agent encounters when navigating the environment with policy $\boldsymbol{\pi}$. There is a one-by-one correspondence between the occupancy measure ρ_π and the policy $\boldsymbol{\pi}$ [1].

GAIL tries to match ρ_{π_E} with generative adversarial networks. A discriminator (or critic) $\mathbf{D}_{\boldsymbol{\psi}}$, parametrized by $\boldsymbol{\psi}$ learns to distinguish expert behavior from non-expert's, while a policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ attempts to emulate the expert. The GAIL objective is given by [15],

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\psi}} \mathbb{E}_{\boldsymbol{\pi}_E} \log \mathbf{D}_{\boldsymbol{\psi}}(\mathbf{x}, \mathbf{u}) + \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}} \log(1 - \mathbf{D}_{\boldsymbol{\psi}}(\mathbf{x}, \mathbf{u})) \quad (14)$$

where $\mathbb{E}_{\boldsymbol{\pi}}$ denotes the expectation over policy π, *i.e.*, the state-action samples $(\mathbf{x}, \mathbf{u})$ are taken from the distribution of occupancy measure ρ_π. This paper deals with observation-action pair $(\mathbf{o}, \mathbf{u})$ instead of state-action pair to avoid state estimation.