Prediction by Anticipation: An Action-Conditional Prediction Method based on Interaction Learning

Ershad Banijamali[1], Mohsen Rohani[1], Elmira Amirloo[1], Jun Luo[1], Pascal Poupart[2]
[1]Noah's Ark Laboratory, Huawei, Markham, Canada
[2]School of Computer Science, University of Waterloo, Waterloo, Canada
{ershad.banijamali1,mohsen.rohani,elmira.amirloo,jun.luo1}@huawei.com, ppoupart@uwaterloo.ca

Abstract

In autonomous driving (AD), accurately predicting changes in the environment can effectively improve safety and comfort. Due to complex interactions among traffic participants, however, it is very hard to achieve accurate prediction for a long horizon. To address this challenge, we propose prediction by anticipation, which views interaction in terms of a latent probabilistic generative process wherein some vehicles move partly in response to the anticipated motion of other vehicles. Under this view, consecutive data frames can be factorized into sequential samples from an action-conditional distribution that effectively generalizes to a wider range of actions and driving situations. Our proposed prediction model, variational Bayesian in nature, is trained to maximize the evidence lower bound (ELBO) of the log-likelihood of this conditional distribution. Evaluations of our approach with prominent AD datasets NGSIM I-80 and Argoverse show significant improvement over current state-of-the-art in both accuracy and generalization.

1. Introduction

Predicting the future state of a scene with moving objects is a task that humans handle with ease. This is due to our understanding about the dynamics of the objects in the scene and the way they interact. Despite recent advancements in machine learning and especially deep learning, teaching machines such understanding remains challenging.

In this paper, we are interested in predicting the high-dimensional observations, e.g. in pixel space, of a scene with multiple interacting agents, where the observations are ego-centric, i.e. the scene is observed from the standpoint of one of the agents (ego-agent). Formally, we consider an action-conditional prediction task, which is defined as finding the next observation from the scene given a sequence of the past and current observations, $\mathbf{o}_{1:t}$, and the ego-agent's action (ego-action) $\mathbf{a}_t$, i.e. optimizing $p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t)$. In particular we consider the prediction task in the challenging framework of autonomous driving (AD), where there is a large number of agents in the scene and they have complicated interactions with each other. The common approach in the literature to tackle this problem is to optimize this conditional probability directly, e.g. [12]. However, since the observations are heavily affected by the ego-actions, these models often fail to generalize to larger set of actions than the training set or to adapt to broader range of environments. Nevertheless, in many applications, specially AD, such generalization is of great importance as handling many critical scenarios require taking actions that are rarely seen in the data collected from real-world driving.

In order to harness the effect of the action, we propose a novel approach based on splitting the original problem into two subproblems where this effect can be learned much easier. In fact, we break the observation into two sets of features: 1) ego-features, $\mathbf{o}_t^{ego}$, which contains the features related to the ego agent (e.g. its position). 2) environment-features, $\mathbf{o}_t^{env}$, which contains all other features than the ego-features including other agents. The action has an effect on both of these feature sets. Our approach is based on the idea that sequences of data that include heavy interactions come from a probabilistic generative process wherein some reacting agents move partly in response to the *anticipated* motion of some acting agents. Therefore, we decompose $p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t)$ into two steps: learning $p(\mathbf{o}_{t+1}^{ego}|\mathbf{o}_{1:t}, \mathbf{a}_t)$ and then learning $p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego})$, i.e. we first learn how the action changes the ego-features and then learn how the environment reacts to this change. However, the effect of the action on the ego-features is much easier to model and often does not need to be learned and can be fixed using the domain knowledge. Then we can learn $p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego})$ from data, which is much easier than learning $p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t)$ since we only have to learn to predict $\mathbf{o}_{t+1}^{env}$ based on the *effect* of action $\mathbf{a}_t$ on the ego-features $\mathbf{o}_{t+1}^{ego}$. Conditioning on $\mathbf{a}_t$ or $\mathbf{o}_{t+1}^{ego}$ is equivalent from an information theoretic perspective, but conditioning on $\mathbf{o}_{t+1}^{ego}$ allows the model to reason about interactions more easily since the ego effect is already

anticipated. Moreover, such conditioning is valid in practice as other agents observe the effect of ego-actions not the actions themselves. Hence, the original action-conditional prediction task to a great extent is reduced to *learning the interaction* of the other agents given the anticipated movement of the ego-agent. For such interaction learning we train a conditional deep generative model, where we combine a deterministic mapping that encodes the history of observations and movement of agents with a stochastic mapping that reasons about the future interactions. By thus recovering the latent generative process, our model is capable of achieving a higher capacity for prediction, i.e. handling a wider range of actions and driving situations.

We consider the observations in the form of occupancy grid maps (OGMs), which minimizes the costly and time-consuming preprocessing (e.g. object detection and tracking). The anticipation step is done through transformations of the original OGM based on motion of the acting vehicle. Since the observations are ego-centric, these transformations fix the observation frame from input to target and therefore the *interaction learning* can be represented by learning the displacement of moving objects conditioned on the next position of the ego-vehicle. This enables us to extend our model to a *difference learning* variant, which works very well in dense slow urban traffic. Our contributions include:

- A novel modular model for multi-step action-conditional prediction for AD is proposed, which factorizes historical sequence data into samples drawn according to an underlying action-conditional distribution that covers a wider range of actions for predictions better than current state-of-the-art models. The method requires no labeling and is scalable with data.
- An extension of the model for difference learning that outperforms the state-of-the-art in dense traffic.
- Experiment results on two prominent AD datasets (NGSIM I-80 and Argoverse) with different interactions among vehicles, i.e. highway and urban area, demonstrating effective coverage of a wide variety of actions and driving situations. An extensive ablative study is conducted to investigate the role of each novel component of our model.

2. Related Work

A large body of literature on prediction tasks in AD is dedicated to prediction in low-dimensional space, i.e. position of the cars in the xy coordinates [15, 6, 18, 2, 5, 1, 33, 28, 13, 25, 26, 27, 3, 34, 17]. In most of these works the prediction task is done by finding the most probable paths for the objects in the environment using generative models. Interaction learning has also been studied in the low-dimensional space using feature-pooling, attention, and graph-based models [14, 16, 22]. However, all of these methods need object detection and tracking (at least at training time), which is computationally expensive and requires labeled data. Moreover, any error in the object detection and tracking can affect the whole system and result in catastrophic failure.

Unsupervised prediction of OGMs [8, 31] has also been studied recently. These methods do not model the effect of action in prediction and thus fail to capture the interactions. In [19] and its extension [7] recurrent neural network (RNN)-based models were employed for OGM prediction. But, both models need data labeling and object detection. Authors in [21] proposed a model for multi-step prediction of OGMs that produces state-of-the-art results on the KITTI dataset[9]. By *removing* the ego-motion from the whole sequence, all the frames are mapped to a reference frame in which the ego-vehicle is frozen, i.e. the global location of the car is fixed. This way, only moving objects in the scene change their locations. There are two major differences between this work and ours. First of all, their predictions are not action-conditional. Secondly, since we *compensate* the ego-motion step-by-step the global location of the ego car is not fixed. Therefore, in contrast to [21], our model can predict for much longer horizons.

In [23, 24, 10] OGM prediction is used for path planning. However, in [23, 24] only one object type (human) is considered. [10] relies on object detection and the OGMs are updated using object models. Model-predictive policy with uncertainty regularization (MPUR) [12], is a state-of-the-art prediction and planning approach in this area. Although the model is successful in predicting the effect of existing actions in the training data, in the case of extreme actions it fails to predict a valid OGM. Alternatively, in [1] synthetic extreme actions are added to the training data in order to handle rare scenarios. However, since this is not a theoretically principled way for generalization, the performance of the algorithm is still limited by actions directly observed in the data. Moreover, addition of random actions not grounded in real interaction contexts can result in invalid scenarios that do not happen in real life, as the other cars do not react to the augmented actions. Finally, the method is an object tracking method with the aforementioned problems.

3. Prediction by Anticipation

The prediction task is described as follows. Given a set of t observations from the scene, denoted by $\mathbf{o}_{1:t}$, and a set of k actions denoted by $\mathbf{a}_{t:t+k-1}$, predict the future k observations, $\mathbf{o}_{t+1:t+k}$. Actions are two-dimensional, which include acceleration, α, and rotation of the steering wheel, τ, $\mathbf{a}_t = [\alpha_t, \tau_t]$. We try to solve this task by maximizing the conditional likelihood $p(\mathbf{o}_{t+1:t+k}|\mathbf{o}_{1:t}, \mathbf{a}_{t:t+k-1})$. The observations include (a) A bird's-eye view (BEV) image, denoted by $\mathbf{i}_t$ for time step t, in the form of an OGM with fixed position, e.g. in the middle of the image, for the ego-vehicle. (b) Position and velocity of the ego-vehicle in each

direction, which are denoted by $\mathbf{p}_t$ and $\mathbf{v}_t$, respectively, and are referred to as *measurements*. These two parts of the observation are both sensory data from the vehicle. However, we focus on predicting the images, as the position and velocity can be deterministically computed given the actions, as described in the next sections.

3.1. Base model

Let's assume $\mathbf{o}_t = \{\mathbf{o}_t^{ego}, \mathbf{o}_t^{env}\}$, where $\mathbf{o}_t^{ego}$ includes the features related to the ego-agent (ego-vehicle), i.e. $\mathbf{p}_t$, $\mathbf{v}_t$, and parts of the OGM related to the ego-vehicle, denoted by $\mathbf{i}_t^{ego}$. Therefore $\mathbf{o}_t^{ego} = \{\mathbf{p}_t, \mathbf{v_t}, \mathbf{i}_t^{ego}\}$. The environment-features, $\mathbf{o}_t^{env}$, include all other features in the observation than the ego-features. In our application this includes parts of the OGM that represent other objects in the scene, e.g. other agents, maps, static objects, etc. Therefore, $\mathbf{o}_t^{env} = \{\mathbf{i}_t^{env}\}$. The actions of the ego-vehicle (ego-actions) affect both ego- and environment-features. The first-order effect of the action is on the ego-features, which can be determined using our *prior knowledge* about the dynamics of the ego-vehicle. The second-order effect is on the environment-feature that should be learned from data. Thus, to maximize the conditional log-likehihood of $\log p(\mathbf{o}_{t+1:t+k}|\mathbf{o}_{1:t}, \mathbf{a}_{t:t+k-1})$, we first split it into k autoregressive steps where at each time step the previous prediction is fed-back to the model. Then for each time step we factorize the log-likelihood into two terms:

$$\begin{aligned}\log p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t) = &\log p(\mathbf{o}_{t+1}^{ego}|\mathbf{o}_t, \mathbf{a}_t) \\ &+ \log p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego}). \end{aligned} \quad (1)$$

Note that to determine the next ego-features, we only need the current observation and action. For the second term, we assume $p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego}, \mathbf{a}_t) = p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego})$. This assumption is based on the fact that the environment does not observe $\mathbf{a}_t$ directly, rather the other agents only observe the *effect* of the ego-action. The actions that we use for training are extracted from the real data and reflect expert's behaviour given the environment's evolution. Therefore we can only focus on learning the effect of ego-actions on the environment and not the opposite. It is also worth mentioning that learning $p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego})$ does not mean that we assume causality between $\mathbf{o}_{t+1}^{env}$ and $\mathbf{o}_{t+1}^{ego}$. We simply learn a distribution where there is a non-causal correlation between $\mathbf{o}_{t+1}^{env}$ and $\mathbf{o}_{t+1}^{ego}$. The idea is that other agents anticipate $\mathbf{o}_{t+1}^{ego}$ and act accordingly. The conditional distribution $p(\mathbf{o}_{t+1}^{env}|\mathbf{o}_{1:t}, \mathbf{o}_{t+1}^{ego})$ models the noise in this anticipation. But since there is a strong correlation between $\mathbf{o}_{t+1}^{env}$ and $\mathbf{o}_{t+1}^{ego}$, then conditioning on $\mathbf{o}_{t+1}^{ego}$ will be very informative.

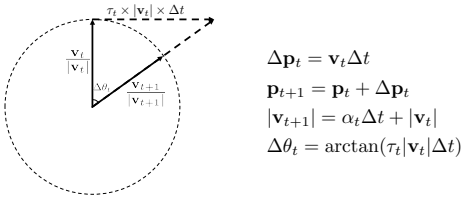


Figure 2: Computing the effect of actions on the future position, velocity and change in the direction of the car.

To implement this idea, we propose a model that consists of two sets of modules (Fig. 3a): 1) Rule-based modules implement the deterministic part of Eq. 1, based on the prior knowledge. 2) Learning-based (prediction) module learns the interactions among the agents.

3.1.1 Rule-based modules

These modules are responsible for factorization according to the presumed underlying generative process. Deterministic functions and transformations are applied on measurements and OGMs, respectively, to account for the effect of actions.
Measurements estimator module: Given the actions and measurements at each time step, computes the next measurements as well as translation and rotation matrices, denoted by $\delta_t = [\Delta\mathbf{p}_t, \Delta\theta_t]$, for the OGM processing modules:

$$\mathbf{p}_{t+1}, \mathbf{v}_{t+1}, \delta_t = f_m(\mathbf{p}_t, \mathbf{v}_t, \mathbf{a}_t), \quad (2)$$

An example of $f_m(.)$ that has been used in literature [12] can be seen in Fig. 2. **Input OGM transformation modules (IOTs):** In order to apply the first-order effect of the action in the OGM, we change the position of the ego-vehicle in the current OGM, $\mathbf{i}_t$. This transformation takes the ego-vehicle to its *anticipated* position at time $t+1$ based on the action. We denote the module that performs this transformation by IOT1, and the output is denoted by $\mathbf{j}_{t+1}^{ego}$: $\mathbf{j}_{t+1}^{ego} = \text{IOT1}(\mathbf{i}_t, \delta_t)$.

Moreover, at each time step of training the target OGM, $\mathbf{i}_{t+1}$, is transformed to account for the effect of the action by reversing the ego-vehicle motion at that point. That is, we transform the whole target OGM in a way that the ego-vehicle has the same position as in $\mathbf{j}_{t+1}^{ego}$. We denote the module for this transformation and its output by IOT2 and $\mathbf{j}_{t+1}^{env}$, accordingly: $\mathbf{j}_{t+1}^{env} = \text{IOT2}(\mathbf{i}_{t+1}, \delta_t)$.

Fig. 1 shows the output of these two modules for a sequence of current and target OGMs. Note that after these

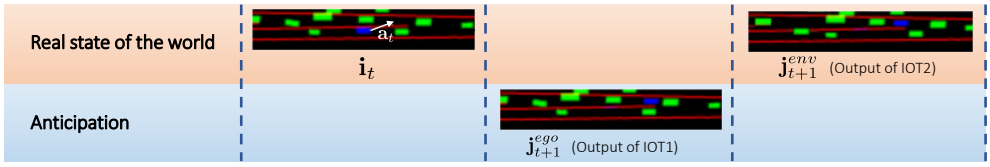


Figure 1: Applying the effect of action on the OGMs: Left image: State of the world in OGM $\mathbf{i}_t$ and corresponding action at time t. Middle image: Anticipated position of the ego-vehicle as the output of IOT1, $\mathbf{j}_{t+1}^{ego}$. Right image: Output of IOT2: transformed target OGM ($\mathbf{i}_{t+1}$) that shows the displacement of other agents given the position of ego-vehicle in $\mathbf{j}_{t+1}^{ego}$.

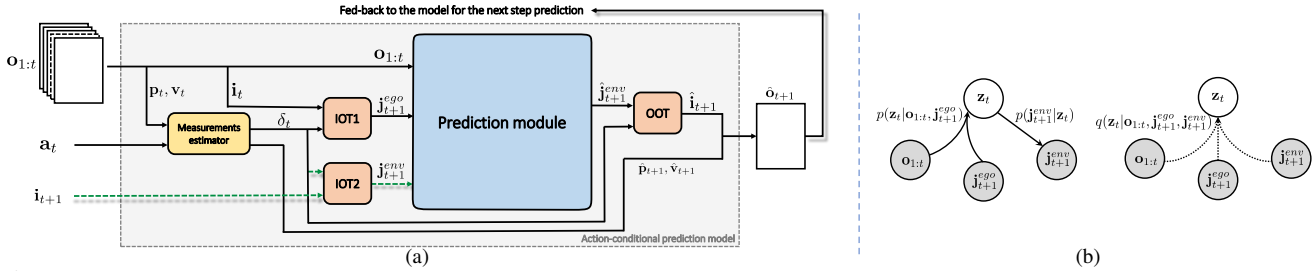


Figure 3: (a) The prediction model with all of its components. The measurement estimation module updates the position and velocity of the ego-vehicle and provides transformation parameters for OGM transformer modules. IOT1 and IOT2 provide information about the first and second-order effect of action on the OGM, respectively. Prediction module is trained to minimize the cost in Eq. 4. (b) Graphical model at time t: Left: Generative links, $p(.)$. Right: Variational links, $q(.)$. Observable variables are gray.

two transformations, only the position of the moving objects will be different in $\mathbf{j}^{ego}_{t+1}$ and $\mathbf{j}^{env}_{t+1}$, while map information and structure of the fixed objects in the OGM, e.g. buildings, tree, parked cars, etc., will be the same. Both $\mathbf{j}^{ego}_{t+1}$ and $\mathbf{j}^{env}_{t+1}$ are fed to the prediction module during training.

Note that we use $\mathbf{j}$ notation to denote the OGMs after the applying IOTs, while $\mathbf{i}$ denotes the ego-centric OGMs. So, the original goal of optimizing for the stochastic mapping $p(\mathbf{i}^{env}_{t+1}|\mathbf{o_{1:t}}, \mathbf{i}^{ego}_{t+1})$ is equivalent to optimizing for $p(\mathbf{j}^{env}_{t+1}|\mathbf{o_{1:t}}, \mathbf{j}^{ego}_{t+1})$, up to some deterministic transformations.

Output OGM transformation module (OOT): This module takes the predicted frame and transforms the whole frame using δ_t such that the ego-vehicle goes back to its fixed position in $\mathbf{i}$ OGMs and environment-features change accordingly. The output, which ideally should be $\mathbf{i}_{t+1}$, is fed back to the model for the next step prediction. This module is necessary to close the loop for *multi-step* prediction.

3.1.2 Learning-based module: Prediction module

The prediction module is the core of our model that predicts how the environment partially reacts to the ego-action, i.e. learns $p(\mathbf{j}^{env}_{t+1}|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$. Using the IOT1 and IOT2 modules the geometry of the target frame, $\mathbf{j}^{env}_{t+1}$, remains the same as the input frame, $\mathbf{j}^{ego}_{t+1}$, at each time step, *regardless of the ego-action*. Therefore $p(\mathbf{j}^{env}_{t+1}|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$ is a smoother function than the original objective function, $p(\mathbf{o}_{t+1}|\mathbf{o}_{1:t}, \mathbf{a}_t)$. Thus, $p(\mathbf{j}^{env}_{t+1}|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$ is intuitively easier to learn. Despite this simplification, the two objectives have the same optimum point, i.e. maximizing one leads to maximizing the other. The first term in Eq. 1 is deterministic and can be removed from the optimization. Also, $\mathbf{j}^{env}_{t+1}$ is uniquely determined by the action $\mathbf{a}_t$ (given $\mathbf{i}_{t+1}$). Consequently, we can re-write the second term of Eq. 1 as $\log p(\mathbf{j}^{env}_{t+1}|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$.

Bottleneck conditional density estimation: We maximize the conditional log-likelihood $\log p(\mathbf{j}^{env}_{t+1}|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$ in the framework of variational Bayes. We build our model upon a Bottleneck Conditional Density Estimation (BCDE) [29] model, a special variant of the conditional variational autoencoders (CVAEs) [30]. The latent code in BCDE acts as bottleneck of information and not just a source of randomness. The prior on the latent variable in BCDE is conditioned on the input. Such conditioning makes the model less prone to overfitting as it allows learning the distribution of the latent code conditioned on input, which is especially helpful for prediction with large horizon. We consider the graphical model in Fig. 3b at each time step for this prediction task. According to our definition of the approximating variational distribution in the graphical model, and also considering $\mathbf{z}_t$ as an information bottleneck between the input, $\mathbf{o}_{1:t}$ and $\mathbf{j}^{ego}_{t+1}$, and the target, $\mathbf{j}^{env}_{t+1}$, the ELBO to be maximized will have the following form:

$$\log p(\mathbf{j}^{env}_{t+1}|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}) \geq \mathbb{E}_{q^*(\mathbf{z}_t)}[\log p(\mathbf{j}^{env}_{t+1}|\mathbf{z}_t)] - \mathrm{KL}\big(q^*(\mathbf{z}_t)||p(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})\big), \quad (3)$$

where $q^*(\mathbf{z}_t) = q(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1})$. We implement each of the conditional probability distributions in Eq. 3 using a neural network and denote the parameters of $p_\psi(.)$ and $q_\phi(.)$ by ψ and ϕ, respectively.

Reconstruction loss and structural similarity: The first term in the ELBO in Eq. 3, can be interpreted as a reconstruction loss in the pixel space that measures the difference between the target OGM, $\mathbf{j}^{env}_{t+1}$, and predicted OGM, $\hat{\mathbf{j}}^{env}_{t+1}$, and we denote it by $\mathcal{D}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1})$. Depending on the type of values in the OGM being continuous or binary we consider Gaussian (with identity covariance matrix) or Bernoulli distributions for the output and replace $\mathcal{D}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1})$ with the mean squared error (MSE) or the cross entropy (CE), respectively. We also add an auxiliary term to our reconstruction loss that computes the Structural Similarity Index (SSIM) [32] loss between prediction and target. The experiments show the effectiveness of adding this term in improving the quality of the predicted frames. The weight of the SSIM term, λ, is set using the validation set. The final training objective to be minimized is:

$$\mathcal{L}_t = \underbrace{\mathcal{D}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1}) + \lambda\big(1 - \mathrm{SSIM}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1})\big)}_{\mathcal{L}^{\text{rec.}}_t} + \underbrace{\mathrm{KL}\big(q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1})||p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})\big)}_{\mathcal{L}^{\text{KL}}_t}. \quad (4)$$

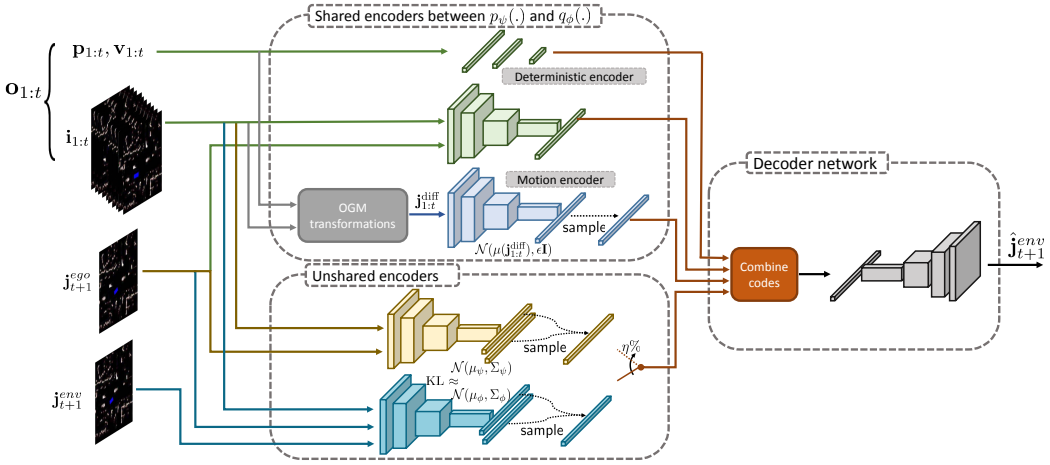


Figure 4: The prediction module at the training time. The measurements are encoded using fully-connected networks, while we use convolutional neural networks to encode and decode OGMs. Reparametrization trick is used for the sampling steps.

For a multi-step prediction with horizon k a summation over $\mathcal{L}_t$ is minimized: $\min_{\psi,\phi} \sum_{j=0}^{k-1} \mathcal{L}_{t+j}$.

Code splitting and sampling from prior: The second term, is a KL divergence regularization that minimizes the distance between the output distributions of $p_\psi(.)$ and $q_\phi(.)$ encoders. Since the observations are highly dynamic with many objects in the scene, merely minimizing the KL divergence does not provide a proper training for the $p_\psi(.)$ encoder. Therefore we employ two ideas to better match these two distributions.

1) We split the latent code $\mathbf{z}_t$ into two parts with two different sets of encoders. For the first part we do not use the target frame as the input of the $q_\phi(.)$ encoder and therefore its parameters can be shared with the $p_\psi(.)$ encoder, which guarantees the minimization of KL divergence. We call this part, *shared code* and since it only encodes the previous and current observations we assume it is partly deterministic. For the second part, called *unshared code*, we assume Gaussian distributions for both the conditional prior $p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$ and the variational posterior $q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1})$ and minimize their KL divergence. This part of $\mathbf{z}_t$ encodes information about the target, $\mathbf{j}^{env}_{t+1}$, and its stochasticity represents the uncertainty about the future. By combining this splitting idea with the BCDE model we introduce another important difference with the vanilla CVAE model, i.e. instead of concatenating the stochastic code with our high-dimensional input, we concatenate it with an encoded version of the input that keeps only useful information. This makes the structure of the decoder simpler with far fewer parameters, and therefore easier to train.

2) To make sure that the encoder of $p_\psi(.)$ for the unshared code is trained properly, we randomly switch between the stochastic samples of the $p_\psi(.)$ and $q_\phi(.)$ encoders, i.e. $\eta\%$ of the time the samples are drawn from $p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})$ instead of $q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1})$. This way the $p_\psi(.)$ encoder is trained by backpropagating both errors of the KL term and the reconstruction term. In our experiments we set $\eta = 10$.

3.2. Difference Learning (DL)

After the OGM transformations, the difference between $\mathbf{j}^{ego}_{t+1}$ and $\mathbf{j}^{env}_{t+1}$ is only in the position of the moving objects. Therefore we also propose a variant of our model that explicitly learns the difference between the two OGMs, denoted by $\mathbf{j}^{\text{diff}}_{t+1} = \mathbf{j}^{env}_{t+1} - \mathbf{j}^{ego}_{t+1}$. In fact, $\mathbf{j}^{\text{diff}}_{t+1}$ represents the motion of other agents in the pixel space. Fig. 5 shows the difference learning module. One issue with this model is that, for incorrect predictions during training, adding $\hat{\mathbf{j}}^{\text{diff}}_{t+1}$ to the input frame $\mathbf{j}^{ego}_{t+1}$ causes $\hat{\mathbf{j}}^{env}_{t+1}$ to go out of the range of the input. This is especially problematic for multi-step prediction when this error is accumulated. A similar structure has been suggested in [21] for predicting binary OGMs, where a classifier layer is used after the summation to take the result within the input range. Although this can potentially resolve the issue, adding such classifier makes the learning process very slow. This is because the classifier ideally acts as a (sigmoid-shaped) clipper and the derivative of the clipper for the out of range values, caused by the actual error, is very small. Instead, in our model we keep the first term of our reconstruction loss the same as before and for the SSIM part we clip the prediction $\hat{\mathbf{j}}^{env}_{t+1}$ and then compute the SSIM:

$$\mathcal{L}^{\text{rec.}}_t = \mathcal{D}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1}) + \lambda\big(1 - \text{SSIM}(\mathbf{j}^{env}_{t+1}, \mathit{clip}(\hat{\mathbf{j}}^{env}_{t+1}))\big). \quad (5)$$

This enables us to backpropagate the incorrect difference predictions through the $\mathcal{D}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1})$ term and also to make use of the SSIM term. For future time-steps the clipped output is fed back to the network.

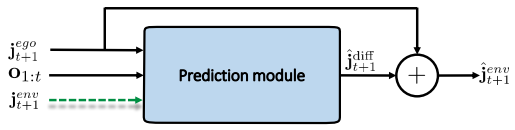


Figure 5: Difference learning module

Motion Encoding: To further enhance the predictive power of our model we capture the past motion of other agents in the scene by encoding the difference between consecutive OGMs in the input sequence. In fact, we built a sequence of $\mathbf{j}^{\text{diff}}_{1:t}$ from the input observations and encode it as a part of the shared code. While $\mathbf{i}_{1:t}$ contain information about motion of other agents, $\mathbf{j}^{\text{diff}}_{1:t}$ highlight such motion relative to the ego-vehicle and therefore encoding $\mathbf{j}^{\text{diff}}_{1:t}$ allows reasoning about higher level motion features such as intention of other agents. We model these features by a Gaussian distribution $\mathcal{N}(\mu(\mathbf{j}^{\text{diff}}_{1:t}), \epsilon\mathbf{I})$, where $\mu(\mathbf{j}^{\text{diff}}_{1:t})$ is a neural network and ϵ is a constant ($\epsilon = 0.5$ in the experiments). Motion encoding is used in both the base model and the DL variant.

Fig. 4 provides a high level architecture of the model. The implementation details of the prediction module are provided in the supplementary material.

4. Experiments

In this section, we evaluate the performance of our model, i.e. prediction by anticipation and its difference learning extension, referred to as PA and PA-DL, respectively.

Baselines: The proposed model is an OGM-in OGM-out model and therefore we compare its performance with two similar prediction models, which are, to the best of our knowledge, the state-of-the-art for this unsupervised prediction task:

- **Forward Model in Model-Predictive Policy Learning with Uncertainty Regularization (FM-MPUR)** [12]: FM-MPUR is a CVAE-based model, which aims to directly maximize the log-likelihood $\log p(\mathbf{o}_{t+1:t+k}|\mathbf{o}_{1:t}, \mathbf{a}_{t:t+k-1})$. Moreover, latent code of the FM-MPUR model has an unconditioned prior. Therefore, latent samples are independent of the input frames. This can potentially hurt the prediction accuracy for longer horizons. The code for this model is publicly available: https://github.com/Atcold/pytorch-PPUU.
- **RNN-based model with Difference Learning component (RNN-Diff)** [21]: RNN-Diff is an encoder-decoder structure that uses an RNN in the code space. Encoding and decoding are done using convolutional layers. The main idea in RNN-Diff is removing the ego-actions for a whole sequence of frames as if the ego-vehicle does not move and the scene is observed by a fixed observer for the whole sequence. Therefore they can just focus on predicting the movement of dynamic objects in a fixed scene by learning the difference between consecutive frames. We use the best architecture of their model, named RNN-Diff2.1, for comparison. We received the code for this model from the authors.

FM-MPUR takes 20 frames as its input. However, we found 10 input frames to be as rich as 20 in terms of information about the past, i.e. the Markov property of the sequence is preserved by 10 frames. Therefore a sum over the conditional log-likelihood can result in a log-likelihood of the whole training set. The RNN-Diff2.1 model also uses 10 input frames. We use sliding window method for training the models.

Metrics: For real-valued OGMs we report *MSE*. For binary OGMs we assign class 1 (positive) to occupied pixel and class 0 (negative) to free pixels and report the results in terms of classification scores, i.e. *true positive rate (TPR) and true negative rate (TNR)*. This enable us to distinguish between the accuracy of predicting occupied and free pixels. TPR is the more important metric for safe driving as it shows how well a model predicts the obstacles in the environment. We also report the results in terms of *average log-likelihood (ALL)*. In fact, we use kernel density estimation (KDE) by approximating the pdf of the training data using 10K training samples and then evaluating the approximated pdf on the predicted frames of the test sequences with different prediction horizons. We use Gaussian kernel with $\sigma = 0.1$.

4.1. Prediction under different driving situations

We ran our experiments on two complimentary datasets, one high-speed highway traffic and the other low-speed dense urban traffic. These two datasets cover a large set of real-life driving scenarios.

NGSIM I-80 dataset: The Next Generation Simulation program's Interstate 80 (NGSIM I-80) [11] dataset consists of 3 batches of 15-minute of recordings from traffic cameras mounted over a stretch of a highway in the US. Driving behaviours are complex with complicated interactions between vehicles moving at high speed. This makes the future state difficult to predict. We follow the same preprocessing proposed in [12] to make the datasets. The images are real-valued RGB with size 117×24. The ego-vehicle is in the center of the blue channel. Other (social) vehicles are in the green channel, which can also be interpreted as the OGM. The red channel has the map information, e.g. lanes.

We train each model using two batches of the 15-minute recording and test it on the third batch and repeat this process three times to cover all combinations. For both PA and PA-DL, we put $\lambda = 0.05$ in the reconstruction cost. Results are shown in Tables 1 and 2. Since the speed of the ego-vehicle is high in this dataset, removing the ego-motion for the whole sequence to train the RNN-Diff2.1, significantly reduces the size of meaningful pixels in the input and target frames and make them practically unusable for training a multi-step model. Therefore the reported results are from a model trained for single-step prediction. This is why its performance dramatically drops for larger values of k. As we can see, PA and PA-DL outperform FM-MPUR. Due to high-speed driving of agents, their positions have dramatic changes from one frame to the next one, in many cases.

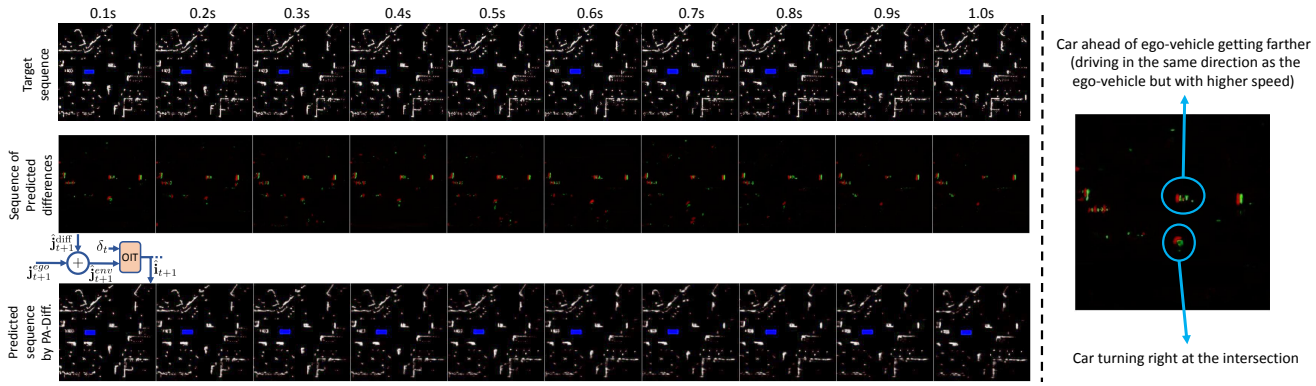


Figure 6: Left: OGM prediction of PA-DL for the Argoverse dataset. Top row shows the target sequence. Middle row shows the sequence of predicted differences, learned by the model, where red areas (negative values) are erased from the frame and green areas (positive values) are added to build the next frame. Bottom row shows the final predicted frame. We demonstrate the mechanism to build the predictions for the first time step. Right: Zoomed-in first predicted difference. Difference learning allows reasoning about the motion of other agents.

Dataset →	NGSIM I-80				Argoverse							
Method	MSE				TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
	$k=1$	$k=5$	$k=10$	$k=20$	$k=1$		$k=5$		$k=10$		$k=20$	
FM-MPUR	3.4 ± 0.1	4.7 ± 0.2	5.2 ± 0.2	7.8 ± 0.1	96.24	99.68	88.44	97.12	74.62	94.21	65.60	85.31
RNN-Diff2.1	4.2 ± 0.2	14.5 ± 0.3	36.6 ± 1.1	80.2 ± 2.0	99.21	**99.91**	93.19	**99.85**	87.23	99.27	80.55	93.92
PA	$\mathbf{2.9 \pm 0.2}$	$\mathbf{4.1 \pm 0.1}$	$\mathbf{4.7 \pm 0.1}$	$\mathbf{6.2 \pm 0.2}$	99.18	99.89	94.12	99.76	90.14	99.48	83.17	96.87
PA-DL	3.3 ± 0.1	4.6 ± 0.1	5.1 ± 0.2	6.7 ± 0.3	**99.40**	**99.91**	**97.13**	99.83	**92.55**	**99.71**	**87.98**	**98.02**

Table 1: Comparison of different models in terms of MSE for NGSIM I-80 and TPR/TNR for Argoverse. For this table predictions for all methods except RNN-Diff2.1 are generated using the mean value for the latent code.

Dataset →	NGSIM I-80				Argoverse			
Method	$k=1$	$k=5$	$k=10$	$k=20$	$k=1$	$k=5$	$k=10$	$k=20$
FM-MPUR	212.6 ± 5.3	207.1 ± 6.9	201.4 ± 5.1	187.3 ± 7.5	624.3 ± 8.1	609.5 ± 5.6	538.4 ± 6.2	461.7 ± 4.5
RNN-Diff2.1	194.2 ± 7.1	141.3 ± 3.6	71.1 ± 3.4	26.6 ± 1.1	656.2 ± 2.9	631.0 ± 3.0	603.1 ± 5.6	545.6 ± 5.8
PA	$\mathbf{236.9 \pm 2.7}$	$\mathbf{232.66 \pm 3.9}$	$\mathbf{225.4 \pm 2.4}$	$\mathbf{211.2 \pm 5.2}$	661.1 ± 6.7	657.1 ± 7.9	635.2 ± 5.3	590.7 ± 4.2
PA-DL	221.7 ± 1.7	217.5 ± 3.1	210.2 ± 4.8	202.9 ± 5.1	$\mathbf{674.6 \pm 2.2}$	$\mathbf{660.2 \pm 2.2}$	$\mathbf{642.2 \pm 3.9}$	$\mathbf{603.4 \pm 3.7}$

Table 2: Comparison of different models in terms of ALL.

Consequently, PA performs slightly better than PA-DL.

Argoverse dataset: For the urban area driving, the OGM sequences are obtained from the Argoverse raw dataset [4]. The dataset contains many different actions and maneuvers, e.g. stops and turns, in slow pace. The LiDAR point-clouds, collected at 10Hz, are converted to BEV 256×256 binary OGMs using ground removal proposed in [20]. We compare the performance of the models in terms of TPR/TNR and ALL in Tables 1 and 2, respectively. Argoverse dataset has more complicated OGM structures and, unlike NGSIM I-80, the ego-motion is usually small and consecutive frames have slight differences. Therefore, PA-DL outperforms PA, and both PA and PA-DL outperform FM-MPUR significantly.

PA-DL and RNN-Diff2.1 perform closely for short-horizon predictions. Again, since in RNN-Diff2.1 the environment is observed from a fixed point, when k is large the dynamic objects eventually leave the scene and the predictions deviate from the actual ground truth. In PA-DL, the difference learning is done step-by-step. As we can see, the performance gap between RNN-Diff2.1 and PA-DL enlarges as k grows. Fig. 6 shows a sample sequence of the Argoverse dataset as well as the outputs of the PA-DL model.

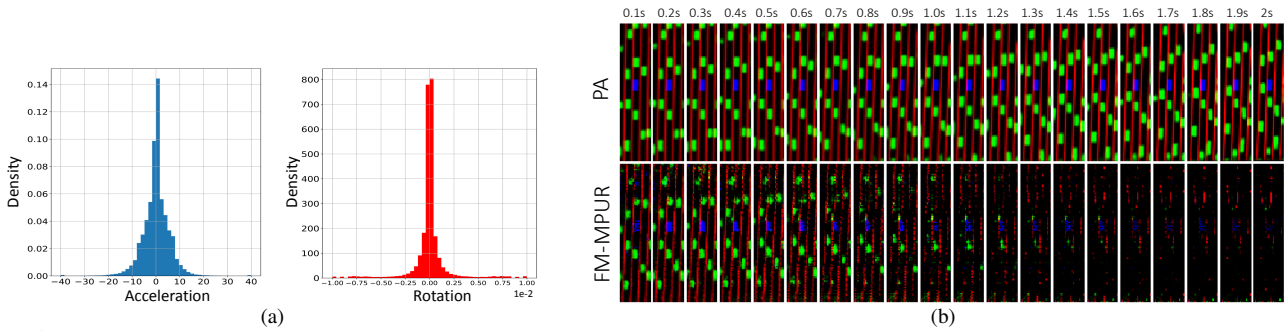
The results of Tables 1 and 2 show that the proposed models outperform the two baselines with a significant margin. Moreover, each of the baselines fails in one of the two datasets, while PA and PA-DL perform the prediction task successfully for both datsets, i.e. both driving situations.

4.2. Prediction for rare actions

In this section we study the performance of the PA and PA-DL algorithms in the presence of rare actions and investigate the effectiveness of employing prior knowledge in providing robustness against the actions that are rare in the training data. Specifically we consider the NGSIM I-80 dataset.

We use the trained models with each of the batches of 15-minute recordings and apply actions that are rarely seen in the training set but are still in the maneuverability range of vehicles. We use the distributions shown in Fig. 7a to sample these actions and apply them to randomly selected sequences of the test set and predict for different prediction horizons. For comparison, we use the FM-MPUR algorithm. Since there is no ground truth, we only report the ALL results.

Table 3 summarizes the evaluation results. It shows that our models significantly outperform FM-MPUR for this task, especially for longer prediction horizons. This suggests that learning environment-features based on the anticipated ego-features makes the prediction task easier to learn for our model, which supports our initial intuition. In fact, by apply-



(a) (b)

Figure 7: (a) Distribution of actions. (b) Effect of constantly applying rare actions on the prediction of PA and FM-MPUR models.

Method	ALL			
	$k = 1$	$k = 5$	$k = 10$	$k = 20$
FM-MPUR	197.3 ± 6.2	175.7 ± 5.1	112.1 ± 7.7	76.4 ± 4.1
PA	$\mathbf{226.1 \pm 2.3}$	$\mathbf{218.6 \pm 4.1}$	$\mathbf{202.2 \pm 3.8}$	$\mathbf{180.4 \pm 6.6}$
PA-DL	213.4 ± 6.8	200.4 ± 4.9	192.9 ± 3.5	174.8 ± 7.9

Table 3: comparison of predictions of PA, PA-DL, and FM-MPUR using rare actions.

Dataset →	NGSIM I-80				Argoverse							
Method	MSE				TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 1$		$k = 5$		$k = 10$		$k = 20$	
PA (no RBM)	3.2 ± 0.2	4.5 ± 0.2	5.0 ± 0.1	7.2 ± 0.4	97.03	99.74	90.15	97.98	80.12	95.23	72.12	88.63
PA (no BCDE)	3.1 ± 0.3	4.2 ± 0.4	4.7 ± 0.2	6.9 ± 0.3	99.01	99.84	93.20	99.64	88.18	99.22	80.14	92.20
PA (no ME)	3.1 ± 0.2	4.3 ± 0.4	4.8 ± 0.3	6.7 ± 0.2	98.65	99.75	92.51	98.50	86.19	98.98	81.35	92.71
PA-DL (no RBM)	4.1 ± 0.4	6.9 ± 0.3	7.3 ± 0.1	10.8 ± 0.2	95.51	99.83	91.94	97.72	81.25	95.74	70.55	86.30
PA-DL (no BCDE)	3.4 ± 0.2	4.7 ± 0.1	5.6 ± 0.2	8.4 ± 0.2	99.20	99.88	96.95	99.70	90.62	99.51	83.24	91.95
PA-DL (no ME)	3.6 ± 0.2	4.7 ± 0.1	5.4 ± 0.2	8.1 ± 0.3	99.05	99.80	96.54	99.71	91.14	99.59	84.64	94.34

Table 4: Results of ablative study on the contributing factors to the performance of our models. no RBM: a model without rule-based modules. no BCDE: a CVAE-based model with unconditioned prior for the latent code. no ME: a model without motion encoding.

ing the extreme actions, the OGMs change dramatically from one time step to the next one. However, we can compensate this change by applying the anticipated modifications to the target OGM. Fig. 7b shows the result of applying rare actions, $\mathbf{a} = [-25, 0]$, for 20 consecutive steps, which can be identified as a very low-probable action sequence according to the distributions in Fig. 7a. This is equivalent to a hard brake in the middle of the road. As we can see our model can predict almost perfectly, while the FM-MPUR model fails after a few predictions. Location of the nearby social vehicles show that the model has learned the dynamics of the traffic: as the ego-vehicle brakes hard, other vehicles continue to move normally except for the one behind it, which is forced to slow down significantly.

4.3. Ablation study

There are three main factors contributing to the better performance of our model compared to the baselines: 1) Employing prior knowledge using rule-based modules to set up the *anticipation-interaction* training. 2) Employing the bottleneck model that conditions the prior of the latent code on input. 3) Encoding *absolute* motion of other agents. We conduct an ablative study on each of these factors for both PA and PA-DL models and provide the results in Table 4 based on MSE and TPR/TNR. Comparing the results of this table with Table 1, we can see that while all factors are contributing, the rule-based modules contribute more. Since the difference learning is a byproduct of our main idea of, removing the rule-based modules degrades the performance of PA-DL significantly. Also motion encoding plays a slightly more important role than the BCDE (conditioned prior) model. The effect of using conditioned prior becomes more apparent for larger values of k. ALL results for both regular and low-probable action settings are provided in the supplementary materials, which show similar behaviors. We also present an ablation study on the effect of the SSIM term on the reconstruction loss in the supplementary materials.

5. Conclusion

We proposed that an observed interaction sequence can be explained by an underlying generative process wherein some agents act partly in response to the anticipated action of other agents. Based on this view, we factorized the interaction sequence into anticipated action and anticipated partial reaction, thereby setting up an action-conditional distribution. We designed a bottleneck conditional density estimation model to learn the distribution. In comparison to the baselines, our model achieves a higher capacity for prediction: it reaches higher accuracy, it handles rare actions much better, it is able to perform well under different driving situations, including high-speed highway driving and complicated urban navigation. While our experiments are limited to vehicle-vehicle interaction, insofar as the understanding generative process is pertinent, our method may also generalize well to other tasks, such as prediction of pedestrian-vehicle interaction, or to other multi-agent domains. Finally, because our model is action conditional, it can serve as a world-model for many downstream tasks.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 2

[2] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018. 2

[3] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 2

[4] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 7

[5] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 2

[6] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018. 2

[7] Julie Dequaire, Peter Ondrúška, Dushyant Rao, Dominic Wang, and Ingmar Posner. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *The International Journal of Robotics Research*, 37(4-5):492–512, 2018. 2

[8] Alberto Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. *arXiv preprint arXiv:1304.1098*, 2013. 2

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 2

[10] Om K Gupta and Ray A Jarvis. Optimal global path planning in time varying environments based on a cost evaluation function. In *Australasian Joint Conference on Artificial Intelligence*, pages 150–156. Springer, 2008. 2

[11] John Halkias and James Colyar. Ngsim interstate 80 freeway dataset. *US Federal Highway Administration, FHWA-HRT-06-137, Washington, DC, USA*, 2006. 6

[12] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *arXiv preprint arXiv:1901.02705*, 2019. 1, 2, 3, 6

[13] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6272–6281, 2019. 2

[14] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018. 2

[15] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 2

[16] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[17] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11553–11562, 2020. 2

[18] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019. 2

[19] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 2

[20] Isaac Miller and Mark Campbell. A mixture-model based algorithm for real-time terrain estimation. *Journal of Field Robotics*, 23(9):755–775, 2006. 7

[21] Nima Mohajerin and Mohsen Rohani. Multi-step prediction of occupancy grid maps with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10600–10608, 2019. 2, 5, 6

[22] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020. 2

[23] Hiroshi Noguchi, Takaki Yamada, Taketoshi Mori, and Tomomasa Sato. Mobile robot path planning using human prediction model based on massive trajectories. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–7. IEEE, 2012. 2

[24] Takeshi Ohki, Keiji Nagatani, and Kazuya Yoshida. Collision avoidance method for mobile robot considering motion and personal spaces of evacuees. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1819–1824. IEEE, 2010. 2

[25] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018. 2

[26] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 2

[27] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019. 2

[28] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *arXiv preprint arXiv:2001.03093*, 2020. 2

[29] Rui Shu, Hung H Bui, and Mohammad Ghavamzadeh. Bottleneck conditional density estimation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3164–3172. JMLR. org, 2017. 4

[30] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015. 4

[31] EG Tsardoulias, A Iliakopoulou, Andreas Kargakos, and Loukas Petrou. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *Journal of Intelligent & Robotic Systems*, 84(1-4):829–858, 2016. 2

[32] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 4

[33] Ruslan Salakhutdinov Yichuan Charlie Tang. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2015. 2

[34] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 2