Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans

Nachiket Deo, and Mohan M. Trivedi, Fellow, IEEE

Abstract—In this paper, we address the problem of forecasting agent trajectories in unknown environments, conditioned on their past motion and scene structure. Trajectory forecasting is a challenging problem due to the large variation in scene structure, and the multi-modal nature of the distribution of future trajectories. Unlike prior approaches that directly learn one-to-many mappings from observed context, to multiple future trajectories, we propose to condition trajectory forecasts on plans sampled from a grid-based policy learned using maximum entropy inverse reinforcement learning policy (MaxEnt IRL). We reformulate MaxEnt IRL to allow the policy to jointly infer plausible agent goals and paths to those goals on a coarse 2-D grid defined over an unknown scene. We propose an attention-based trajectory generator that generates continuous valued future trajectories conditioned on state sequences sampled from the MaxEnt policy. Quantitative and qualitative evaluation on the publicly available Stanford drone dataset (SDD) shows that our model generates trajectories that are (1) diverse, representing the multi-modal predictive distribution, and (2) precise, conforming to the underlying scene structure over long prediction horizons, achieving state of the art results on the TrajNet benchmark split of SDD.

Index Terms—Multi-modal trajectory forecasting, maximum entropy inverse reinforcement learning

1 INTRODUCTION

Forecasting the motion of humans and human driven vehicles is a useful ability with use cases ranging from path planning for autonomous robots, and building models for simulating pedestrian and driver behavior, to early detection of dangerous situations in surveillance videos. There is inherent uncertainty in predicting the future, making trajectory forecasting a challenging task. However, vision sensors such as cameras and LIDARs, and algorithms for object detection and multi-object tracking capture useful cues to narrow down this uncertainty. The track histories of agents provide motion cues, such as the agent’s speed and direction of motion. Additionally, static scene elements around the agent, such as the locations of roads, sidewalks, terrain, buildings and obstacles, provide useful information. One can infer potential goals of agents, their path preferences and constraints on their motion from static scene elements around them. We address the problem of predicting the future locations of pedestrians and vehicles over a prediction horizon of 10 seconds, conditioned on a snippet of their track history and a bird’s eye view representation of the static scene around them. In particular, we wish to forecast trajectories in unknown environments, where prior observations of trajectories are unavailable.

There are several challenges in forecasting agent trajectories in unknown environments:

- **Inferring goals and path preferences**: Without prior observations of agent trajectories in a scene, potential goals and path preferences of agents need to be inferred using visual data.
- **Forecasts that conform to the scene**: Additionally, the forecast trajectories need to conform to the inferred goals and path preferences of the agents.
- **Variability in scene structure**: Scene elements such as roads, sidewalks, crosswalks and buildings can be found in a variety of configurations. Thus there’s high variability in the inputs to the trajectory forecasting model.
- **Non-linearity of agent trajectories**: Agents can make several decisions over a 10 second prediction horizon, leading to trajectories that can be highly non-linear. Thus, there’s high variability in the outputs of the trajectory forecasting model.
- **Multi-modality**: Finally, the distribution of future trajectories of agents is multi-modal. In any given scene, an agent can have one of multiple potential goals, with multiple paths to each goal. Regression based approaches have been shown to average the modes of the predictive distribution [1], [2], [3]. This would lead to trajectory forecasts that may not conform to the underlying scene.

Recent approaches have addressed multi-modality in trajectory forecasting by learning one-to-many mappings, from available context such as the static scene and past motion of agents, to multiple future trajectories. Some works [3], [4], [5], [6], [7], [8] use mixture models, with a mixture component assigned to each mode of the predictive distribution. However, this requires the number of modes to be fixed beforehand. Several approaches [1], [2], [9], [10], [11], [12], [13], [14] instead use conditional generative models. Conditional generative models models map input context and a sample from a simple latent distribution, to a trajectory output. These models can be sampled from indefinitely, and output continuous valued trajectories. Conditional generative models and mixture models need to learn a mapping from a high dimensional input space (variable static scene) to a high dimensional output space (continuous valued tra-
Fig. 1: Forecasts generated by P2T: We address the problem of forecasting agent trajectories in unknown environments. The inputs to our model (left) are a 3.2 second snippet of the agent’s past trajectory, and a bird’s eye view representation of the scene around them. Our model infers potential goals of the agents (left-middle) and paths to these goals (right-middle) over a coarse 2-D grid defined over the scene. Finally, it generates continuous valued trajectories conditioned on the grid-based plans over a 10 second horizon (right).

1) Joint inference of goals and paths by learning rewards: We reformulate the maximum entropy inverse reinforcement learning framework to learn transient path state rewards and terminal goal state rewards. Our reformulation allows for joint inference of goals, and paths to goals, in unknown scenes. This alleviates the need for a pre-defined absorbing goal state as per the original MaxEnt IRL formulation.

2) Trajectories conditioned on plans: We refer to state sequences sampled from the MaxEnt policy as plans. We propose an attention based trajectory generator that outputs continuous valued trajectories conditioned on sampled plans, rather than a latent variable. Compared to conditional generative models, our model outputs trajectories that better conform to the underlying scene over longer prediction horizons. Additionally, the state sequences of the MaxEnt policy allow for better interpretability compared to the latent space of a conditional generative model.

In this work, we seek to leverage the transferability of grid based MaxEnt IRL approaches, while allowing for sampling of continuous valued trajectories similar to conditional generative models. We present P2T (Plans-to-Trajectories), a planning based approach to generate long-term trajectory forecasts in unknown environments. In particular, our approach relies on two key ideas.

1. We refer to agent locations without assigned times as paths, and agent locations with assigned times as trajectories.

MDP formulation: We consider a Markov decision process $M = \{S, A, T, r\}$, for a finite horizon setting with $N$ steps. $S$ is the state space consisting of cells in a 2-D grid defined over the scene. $A$ is the action space consisting of 4 discrete actions, $\{up, down, left, right\}$, to move to adjacent cells. We assume deterministic dynamics, where $T : S \times A \rightarrow S$ is the state transition function. Finally, $r : S \rightarrow \mathbb{R}_0^{N}$ is the reward function.
reward function mapping each state to a real value less than or equal to 0. We assume that the initial state \( s_{\text{init}} \) and the goal state \( s_{\text{goal}} \) of the MDP are known.

**MaxEnt IRL objective:** Under the maximum entropy distribution, the probability of observing a state action sequence \( \tau = \{(s_1, a_1), (s_2, a_2), \ldots, (s_N, a_N)\} \) is proportional to the exponential of its reward.

\[
P(\tau) = \frac{1}{Z} \exp \left( \sum_{i=1}^{N} r(s_i) \right),
\]

where \( Z \) the normalizing constant. MaxEnt IRL involves learning a reward function \( r(\tau) \) parametrized by a set of parameters \( \theta \), operating on a set of features extracted for each state \( s \). The objective is to learn a reward function that maximizes the log likelihood of observing a training set of demonstrations \( \tau = \{\tau_1, \tau_2, \ldots, \tau_K\} \)

\[
\arg \max_{\theta} \log L_{\theta} = \arg \max_{\theta} \sum_{\tau \in T} \log \left( \frac{1}{Z_{\theta}} \exp(r(\tau)) \right).
\]

This can be solved using stochastic gradient descent, with the gradient of the log likelihood \( L_{\theta} \) simplifying to

\[
\frac{dL_{\theta}}{d\theta} = \sum_{\tau \in T} (D_{\tau} - D_{\theta}) \frac{dr(\tau)}{d\theta},
\]

where, \( D_{\tau} \) are the state visitation frequencies (SVFs) for the training demonstration \( \tau \) and \( D_{\theta} \) are the expected SVFs for the MaxEnt policy given the current set of reward parameters \( \theta \). If a deep neural network is used to model the reward function \( r(\tau) \), \( d\theta \) can be obtained using backpropagation as described in [22]. \( D_{\theta} \) is obtained using Algorithm 1 and Algorithm 2.

**Approximate value iteration:** Algorithm 1 involves solving for the MaxEnt policy \( \pi_{\theta} \), given the current reward function \( r_{\theta} \) and the goal state \( s_{\text{goal}} \). \( \pi_{\theta} \) represents the probability of taking action \( a \) given state \( s \). The policy can be stationary, i.e., independent of the time step \( \pi_{\theta}(a|s) \), or non-stationary \( \pi_{\theta}^{(n)}(a|s) \). We use a non-stationary policy as in [23], [24]. Algorithm 1 involves iterative updates of the state and action log partition functions \( V(s) \) and \( Q(s, a) \). These can be interpreted as soft estimates of the expected future reward given state \( s \) and the expected future reward given state-action pair \( (s, a) \) respectively. \( V(s) \) is initialized to 0 for \( s_{\text{goal}} \) and \(-\infty\) for all other states. \( V(s) \) and \( Q(s, a) \) are then iteratively updated over \( N \) steps, while holding \( V(s_{\text{goal}}) \) fixed at 0. For each step, \( \pi_{\theta} \) is given by

\[
\pi_{\theta}^{(n)}(a|s) = \exp \left( Q_{\theta}^{(n)}(s, a) - V^{(n)}(s) \right).
\]

Holding \( V(s_{\text{goal}}) \) fixed to 0, while initializing all other \( V(s) \) values to \(-\infty\) ensures that the MDP ends at \( s_{\text{goal}} \).

**Policy propagation:** Algorithm 2 involves calculating the SVFs. It involves repeatedly applying \( \pi_{\theta} \) for \( N \) steps, starting with the initial state distribution, to give SVF at each step. The SVF corresponding to the goal state is set to 0 at each step, since the goal state absorbs any probability mass that reaches it. The expected SVF \( D_{\theta} \) is obtained by summing the SVFs over the \( N \) steps.

### Algorithm 1 Approx. value iteration (goal conditioned)

**Inputs:** \( r_{\theta}, s_{\text{goal}} \)

1: \( V^{(N)}(s) \leftarrow -\infty, \forall s \in S \)

2: for \( n = N, \ldots, 2, 1 \) do

3: \( V^{(n)}(s_{\text{goal}}) \leftarrow 0 \)

4: \( Q^{(n)}(s, a) = r_{\theta}(s) + V^{(n)}(s) \), \( s' = T(s, a) \)

5: \( V^{(n-1)}(s) = \log \sum a \exp( Q^{(n)}(s, a) ) \)

6: \( \pi_{\theta}^{(n)}(a|s) = \exp \left( Q^{(n)}(s, a) - V^{(n)}(s) \right) \)

7: end for

### Algorithm 2 Policy propagation (goal conditioned)

**Inputs:** \( r_{\theta}, s_{\text{init}}, s_{\text{goal}} \)

1: \( D^{(1)}(s) \leftarrow 0, \forall s \in S \)

2: \( D^{(1)}(s_{\text{init}}) \leftarrow 1 \)

3: for \( n = 1, 2, \ldots, N \) do

4: \( D^{(n)}(s_{\text{goal}}) \leftarrow 0 \)

5: \( D^{(n+1)}(s) = \sum_{s', a} \pi_{\theta}^{(n)}(a|s') D^{(n)}(s') \), \( T(s', a) = s \)

6: end for

7: \( D(s) = \sum_{n} D^{(n)}(s) \)

### Path forecasting conditioned on goals:

The MaxEnt policy \( \pi_{\theta}^* \), for the converged reward model \( r_{\theta} \), can be sampled from, to give path forecasts on the 2-D grid from the \( s_{\text{init}} \) to \( s_{\text{goal}} \). Since \( \pi_{\theta}^* \) is stochastic, the policy can explore multiple paths within the scene to the goal state. However, for most cases of pedestrian or vehicle trajectory forecasting, \( s_{\text{goal}} \) is unknown, and needs to be inferred. Additionally, sampling \( \pi_{\theta}^* \) only provides future paths, without mapping them to specific times. A step for the MDP need not correspond to a fixed time interval. Different agents can have different speeds. Agents can also accelerate or decelerate over the course of the 10s prediction horizon.

### 3 PROPOSED APPROACH

As discussed in section 1, we leverage the transferability of grid based MaxEnt IRL, while not requiring knowledge of \( s_{\text{goal}} \), and generate continuous valued trajectories, mapped to specific times in the future. Figure 2 provides an overview of P2T, our proposed approach. P2T consists of three components.

The first component is a reward model, comprised by convolutional and pooling layers. At each cell on a coarse 2-D grid, the reward model maps local scene context and motion features capturing the agent’s track history, to a transient path state reward and a terminal goal state reward. We describe the reward model in greater detail in section 3.2.

The next component is a MaxEnt policy independent of pre-defined goal states. We reformulate MaxEnt IRL to allow for inference of goal and path states, given the path and goal rewards learned by the reward model (see section 3.1). We obtain a single policy that can be sampled to generate paths to different plausible goals on the 2-D grid. We refer to each state sequence sampled from the policy as a plan.

The final component of P2T is an attention based trajectory generator, that outputs continuous valued trajectories conditioned on the sampled plans. The trajectory generator encodes the track history of the agent using a gated recurrent unit (GRU), and the sampled plans using a bidirectional...
MDP formulation:

- **State space:** Potentially any cell location on the 2-D grid could be the goal of the agent, or a point on their future path. We define the state space $S = \{S_p, S_g\}$. $S_p$ is the set of path states and $S_g$ is the set of goal states. Each cell location on the 2-D grid has an associated path state belonging to $S_p$ and a goal state belonging to $S_g$. The policy terminates on reaching any goal state.

- **Action space:** $A = \{\text{up, down, left, right, end}\}$. The up, down, left and right actions allow transitions from path states to adjacent path states. Additionally, we define an end action that transitions the MDP from a path state to the goal state at the same cell location.

- **Transition function:** $T : S_p \times A \rightarrow S$ maps path state and action pairs to other path states and goal states. Since goal states are terminal, the MDP has no transitions out of a goal state.

**Rewards:** We learn two functions, $r_{pg}$ corresponding to path rewards, and $r_{gs}$ corresponding to goal rewards.

Approximate value iteration with inferred goals: Algorithm 3 depicts our modified approximate value iteration, unconstrained on $s_{goal}$. Unlike algorithm 1, we do not hold the $V(s_{goal})$ fixed at 0 to enforce goal directed behavior. Instead, we use $r_{gs}$ to learn a policy that induces a multi-modal distribution over possible goal states. The inputs to algorithm 3 are the learned rewards $(\theta)$. We initialize $V(s) \leftarrow -\infty$ for all path states $S_p$. This is because we want the MDP to end up at some goal state within the $N$ step finite horizon. Since the goal states are terminal, the MDP receives the goal rewards only once. We thus hold $V(s)$ fixed at 0 to enforce goal directed behavior.

We initialize $V(s) \leftarrow -\infty$ for all path states $S_p$. We then iteratively update the state-action log partition function $Q^{(n)}(s, a)$ and the state log partition function $V^{(n)}(s)$ for the path states $S_p$ over $N$ steps. At the end of each step, the MaxEnt policy is obtained by taking the ratio of the exponent of $Q^{(n)}(s, a)$ and $V^{(n)}(s)$, as per equation (4).

Policy propagation with inferred goals: Algorithm 4 depicts policy propagation independent of $s_{goal}$. This is almost identical to algorithm 2. The only difference is, we do not set the goal state SVFs to 0, as in line 4 of algorithm 2. This is because we use the goal SVFs to train the reward model for $r_{gs}$, using equation (3). We use a frame of reference centered at the agent’s location at the time of prediction. Thus, $s_{init}$ is always the path state at the center of the grid.

**Algorithm 3** Approx. value iteration (inferred goals)

1: $V^{(N)}(s) \leftarrow -\infty, \forall s \in S_p$
2: for $n = N, \ldots, 2, 1$ do
3:    $V^{(n)}(s) \leftarrow r_{gs}(s), \forall s \in S_p$
4:    $Q^{(n)}(s, a) = r_{gs}(s) + V^{(n)}(s'), \forall s \in S_p, s' = T(s, a)$
5:    $V^{(n-1)}(s) = \log\sum_{a} e^{Q^{(n)}(s, a)}, \forall s \in S_p$
6:    $\pi_{gs}^{(n)}(a|s) = \exp\left(Q^{(n)}(s, a) - V^{(n)}(s)\right)$
7: end for
3.2 Reward model

We define a reward model consisting purely of convolutional and pooling layers. This allows us to learn a mapping from local patches of the scene to path and goal rewards. The equivariance of the convolutional layers allows the reward model to be transferred to novel scenes with a different configuration of scene elements. Figure 3 shows our reward model. It consists of three convolutional neural networks (CNNs).

CNN_{feat} serves as a scene feature extractor, operating on the birds eye view representation I of the static scene around the agent:

\[ \phi_I = \text{CNN}_{feat}(I). \]  

(5)

The spatial dimensions of the scene features \( \phi_I \) equal the size of the 2-D grid corresponding to our state space \( S \). In addition to scene features, we want our goal and path rewards to depend on the past motion of the agent. Thus, similar to Zhang et al. [18], we concatenate the scene features with feature maps encoding the agent’s motion, and the locations of the grid cells:

\[ \phi_M = [|v|, \Delta \theta, r]. \]  

(6)

Here, \(|v|\) is the speed of the agent. This value is replicated over the entire feature map. \( r \) is the distance of each cell in the grid from the origin of our co-ordinate system, centred at the agent’s location at the time of prediction. Finally, \( \Delta \theta \) is the angular deviation between a cell location and the instantaneous direction of the agent’s motion at the time of prediction.

CNN_p and CNN_g map the scene and motion features to path and goal rewards respectively:

\[ r_{\phi_0} = \text{CNN}_p(\phi_I, \phi_M). \]  

(7)

\[ r_{\phi_0} = \text{CNN}_g(\phi_I, \phi_M). \]  

(8)

Implementation details: To keep image sizes tractable, we downsample images from the Stanford drone dataset by a factor of 5. \( I \) is a 168 \( \times \) 168 crop of the image around the agent’s location at the time of prediction. CNN_{feat} consists of the first two blocks of the VGG16 model [28]. This downsamples the spatial dimension of the feature maps to 42 \( \times \) 42. This is followed by a 2 \( \times \) 2 convolutional layer with depth 32 and stride 2, to aggregate context at each cell location. This gives 32 scene feature maps over a 21 \( \times \) 21 grid. CNN_p and CNN_g have identical architectures consisting of two 1 \( \times \) 1 convolutional layers. The first layer has depth 32, and the second layer has depth 1 to give a single path or goal reward value at each cell. We apply the log-sigmoid activation at the outputs of CNN_p and CNN_g to restrict reward values between \(-\infty, 0\). The reward model is trained to maximize the log-likelihood \( L_\theta \) of agent paths in the train set shown in equation (2), with gradients given by equation (3). The state visitation frequencies \( D_t \) for both path and goal states are obtained using algorithms 3 and 4. We use Adam [27] with learning rate 0.001 to train the model. We augment the training data by including random rotations of \( I \) and modifying \( \phi_M \) accordingly. Additionally, we initialize CNN_{feat} by pretraining for semantic segmentation as a fully convolutional network [28] on the ISPRS Potsdam dataset [29] consisting of satellite images with scenes similar to the Stanford drone dataset.

3.3 Trajectories conditioned on plans

Consider the optimal MaxEnt policy \( \pi^*_\theta \) obtained using algorithm 3 for the converged reward model. Consider \( K \) state sequences or plans \( S^{(i)} \) sampled from \( \pi^*_\theta \), with the \( i^{th} \) plan given by

\[ S^{(i)} = [s_1^{(i)}, s_2^{(i)}, \ldots, s_N^{(i)}]. \]  

(9)

We expect the \( K \) plans to end at a diverse set of goal states, and explore various paths to these goals. Additionally, each plan \( S^{(i)} \) can be expected to conform to the underlying scene and model the agent’s sequential decision making. However, the plans by themselves do not capture the dynamics of the agent’s motion. A fast moving agent can make more progress along a plan compared to a slow moving agent over a fixed prediction horizon \( T_f \). The dynamics of the agent’s motion can be estimated using a snippet of their most recent track history, over time \( T_h \):

\[ X = [X_{-T_h}, \ldots, X_1, X_0], \]  

(10)

where the \( X_i \)'s correspond to past locations of the agent, with the subscript \( t \) representing time. We define \( t = 0 \) at the prediction instant.

We thus seek a model that, for each sampled plan \( S^{(i)} \) and track history \( X \), generates a continuous valued trajectory over a prediction horizon \( T_f \):

\[ Y^{(i)} = [Y_1^{(i)}, Y_2^{(i)}, \ldots, Y_{T_f}^{(i)}], \]  

(11)

where \( Y_t \) is the future location of the agent at time \( t \). We propose a trajectory generator modeled as a recurrent neural
network encoder-decoder, equipped with soft attention\cite{25}. Our model has the following components.

**Motion encoder:** We encode the track history \(X\) using a GRU encoder, where the state of the GRU at time \(t\) is given by

\[
h_{m_t} = \text{GRU}_m(h_{m_{t-1}}, e_x(X_t)). \tag{12}\]

Here \(e_x()\) is a fully connected embedding layer for the track co-ordinates. The GRU state at the prediction instant, \(h_{m_0}\), can be expected to encode the motion of the agent.

**Plan encoder:** We encode a sampled plan \(S^{(i)}\) using a bidirectional GRU (BiGRU) encoder. We use a BiGRU since this allows us to use the soft attention mechanism in the decoder. Figure 4 shows the plan encoder in greater detail. For each state \(s_n\) in a sampled plan \(S^{(i)}\), we first extract two features: (1) The location co-ordinates of the grid-cell corresponding to \(s_n\), and an image crop of the original image \(I\), at the grid cell location. These are then encoded using fully connected layers. The embeddings are concatenated to give the state encoding \(\phi_s(s_n)\). The state of the BiGRU at step \(n\) is given by

\[
h_{s_n}^{(i)} = \text{BiGRU}_s(h_{s_{n-1}}^{(i)}, h_{s_{n+1}}^{(i)}, \phi_s(s_n)). \tag{13}\]

**Attention based decoder:** We use a GRU decoder equipped with a soft attention module to generate the output trajectories \(Y^{(i)}\). Our core idea is to allow the decoder to attend to specific states of the sampled plan \(S^{(i)}\) as it generates trajectories along the plan. Thus, the decoder can attend to just the first few states of sampled plans, as it generates the future trajectories for a slow moving agent. On the other hand, it can attend to later states while generating a fast moving agent’s trajectories.

We initialize the state of the decoder using the final state of the motion encoder,

\[
h_{dec} = h_{m_0}. \tag{14}\]

This provides the decoder a representation of the agent’s motion. The decoder state is then updated over the prediction horizon according to

\[
h_{dec_i} = \text{GRU}_{dec}(h_{dec_{i-1}}^{(i)}, \text{Att}(h_{dec_{i-1}}^{(i)}, h_{s_{1:N}}^{(i)})), \tag{15}\]

where \(\text{Att}()\) is the attention module. Finally, the output trajectory at each time stamp is given by a fully connected layer \(o_y()\) operating on the decoder states

\[
y_t^{(i)} = o_y(h_{dec_i}^{(i)}). \tag{16}\]

**Implementation details:** We use track history of 3.2 seconds and a prediction horizon of 10 seconds. We assume an agent centric frame of reference with \(X_0 = (0, 0)\). We use a 32 sized state vector for each of the GRUs. All fully connected embedding layers for location co-ordinates have size 16. We used a fully connected layer of size 32 to embed image crops along the sampled plans. Our attention module is a multi-layer perceptron (MLP) with one hidden layer of size 32.

To train the model, we sample \(K\) plans for each prediction instance using the MaxEnt policy \(\pi_G\), and generate \(K\) output trajectories \(\{Y^{(1)}, Y^{(2)}, \ldots, Y^{(K)}\}\). We minimize

\[
\text{mADE}_K = \min_{i \in \{1,\ldots,K\}} \frac{1}{T_f} \sum_{t=1}^{T_f} \left\| Y_t^{GT} - Y_t^{(i)} \right\|_2, \tag{17}\]

where \(Y_t^{GT}\) is the ground truth future trajectory of the agent. The mADE loss has been used in prior work for training models for multi-modal trajectory forecasting\cite{2,5,9}. For a model generating multiple trajectories, it avoids penalizing plausible future trajectories that do not correspond to the ground truth. Similar to \cite{2,9}, we use \(K = 20\) for training the model.

To speed up convergence, we pre-train the model to minimize the average displacement error between \(Y^{GT}\) and the trajectory predicted by the model conditioned on the ground truth plan of the agent \(Y^{SC}\).

\[\text{Fig. 4: Plan encoder: For each state in a sampled plan, we encode the local scene path at the grid cell, and the location co-ordinates of the cell and term it } \phi_s(s). \text{ This is then fed into bidirectional GRU to encode the the entire sampled plan. Our GRU decoder generates output trajectories by attending to the plan encoding.}\]

\[\text{the minimum average displacement error (mADE) loss over the training set.}\]

\[\text{where } Y_t^{GT} \text{ is the ground truth future trajectory of the agent.}\]

\[\begin{align*}
\text{mADE}_K &= \min_{i \in \{1,\ldots,K\}} \frac{1}{T_f} \sum_{t=1}^{T_f} \left\| Y_t^{GT} - Y_t^{(i)} \right\|_2, \tag{17}\end{align*}\]

\[\text{where } Y_t^{GT} \text{ is the ground truth future trajectory of the agent.}\]

\[\text{The mADE loss has been used in prior work for training models for multi-modal trajectory forecasting}\cite{2,5,9}.\]

\[\text{For a model generating multiple trajectories, it avoids penalizing plausible future trajectories that do not correspond to the ground truth. Similar to } \cite{2,9}, \text{we use } K = 20 \text{ for training the model.}\]

\[\text{To speed up convergence, we pre-train the model to minimize the average displacement error between } Y^{GT} \text{ and the trajectory predicted by the model conditioned on the ground truth plan of the agent } Y^{SC}.\]

4 **Experimental Analysis**

4.1 **Dataset**

We use the Stanford drone dataset (SDD)\cite{20} for evaluating our model. SDD consists of trajectories of pedestrians, bicyclists, skateboarders and vehicles captured using drones at 60 different scenes on the Stanford university campus. The dataset provides bird’s eye view images of the scenes, and locations of tracked agents in the scene’s pixel co-ordinates. The dataset contains a diverse set of scene elements like roads, sidewalks, walkways, buildings, parking lots, terrain and foliage. The roads and walkways have different configurations, including roundabouts and four-way intersections.

We use the dataset split defined in the TrajNet benchmark\cite{21} and used in prior work\cite{9,10,12}, for defining our train, validation and test sets. The dataset is split based on scenes. Thus, the train, validation and test sets all have different scenes from the 60 total scenes. This allows us to evaluate our model on unknown scenes where it hasn’t seen prior trajectory data. We list the scenes used in the train, validation and test sets in Table 1 here for reference. Each name-number pair (eg. bookstore-2), corresponds to a unique scene.
Fig. 5: Baselines: In addition to prior approaches [1], [2], [9], [10], [12], we consider two additional baselines that can be considered ablations of our approach. First, we consider a conditional generative model, where we drop the grid based plans and just condition future trajectories on samples from a latent distribution. Next we consider grid based plans generated via a behavior cloning model, rather than our MaxEnt IRL formulation.

While all the conditional generative models listed above map a sample from a simple prior distribution to future trajectories conditioned on context (scene and past trajectories of agents), CF-VAE allows the prior distribution itself to be learned conditioned on context.

Additionally, we consider two other baselines (see figure 5) that are ablations of the proposed approach.

**Conditional Generative model (CG):** First, we wish to evaluate the usefulness of conditioning future trajectories on grid-based plans. Our first ablation is a conditional generative model (see figure 5a). The CG baseline directly maps the outputs of CNN_{feat} and GRU_{m} to future trajectories. To allow for multiple trajectory forecasts, we append the inputs to GRU_{dec} with a sample \( z^{(i)} \) from the standard normal distribution. GRU_{dec} attends to the scene features while generating trajectories. The CG model is also trained to minimize the mADE_{K} loss with \( K = 20 \).

**Plans to trajectories-Behavior cloning (P2T_{BC}):** Next, we wish to evaluate the usefulness of using IRL for generating grid-based plans. Thus, we consider a model where the grid based plans are generated using a behavior cloning (BC) policy (see figure 5b). Instead of mapping \((\phi_{I}, \phi_{M})\) to rewards, the behavior cloning model maps \((\phi_{I}, \phi_{M})\) to probabilities of taking each of the 5 actions, at each grid cell, giving the behavior cloning policy \( \pi_{BC} \). The trajectory generator then forecasts trajectories conditioned on behavior cloning plans. We train the behavior cloning model using the cross entropy loss, where the ground truth actions are given by demonstrations \( \tau \) from the train set. We backpropagate the loss only for those states in the grid that are explored by the demonstration \( \tau \).

Finally, we consider the complete proposed model. While reporting results, we refer to the complete proposed model as P2T_{IRL}.

### 4.3 Metrics
For evaluating a trajectory forecasting model, we need a metric for how much the forecasts deviate from the ground truth future trajectory. However, since our model generates forecasts from a multi-modal distribution, we need a metric...
propose to evaluate (and even train) models using two casts was first addressed by Rhinehart.

Poor measures for its ‘precision’. mFDE \( K \) conform to the underlying scene. Thus, while mADE \( K \) ground truth. Thus a model that generates a very diverse trajectories as long as one of the ground truth, they also do not penalize implausible future predicted trajectories that don’t conform to the final prediction horizon at the end of the prediction horizon. The mFDE \( K \) metric is given by equation (17). The mFDE \( K \) is similar to the mADE \( K \) metric, but only considers the prediction error for the final predicted location at the end of the prediction horizon. The mFDE \( K \) metric is given by

\[
mFDE_K = \min_{i \in \{1,...,K\}} \| Y_{GT}^{(i)} - Y_{T_f}^{(i)} \|_2, \tag{18}
\]

where \( T_f \) is the prediction horizon.

While the mADE \( K \) and mFDE \( K \) errors avoid penalization of plausible future trajectories that don’t conform to the ground truth, they also do not penalize implausible future trajectories as long as one of the \( K \) trajectories is close to the ground truth. Thus a model that generates a very diverse set of \( K \) trajectories by random guessing can achieve low mADE \( K \) and mFDE \( K \) values, even if the trajectories do not conform to the underlying scene. Thus, while mADE \( K \) and mFDE \( K \) serve as good measures of the ‘recall’ of the model for the multi-modal predictive distribution, they serve as poor measures for its ‘precision’.

This trade-off between diversity and precision of forecasts was first addressed by Rhinehart et al. [11]. They propose to evaluate (and even train) models using two KL divergence measures \( H(p, q_p) \) and \( H(q_\pi, \bar{p}) \). \( H(p, q_p) \) measures the likelihood of the ground truth trajectory \( p \) under a model’s predictive distribution \( q_\pi \). \( H(p, q_\pi) \) serves a role similar to the mADE \( K \) and mFDE \( K \) metrics, as a measure of the ‘recall’ of the model. On the other hand, \( H(q_\pi, \bar{p}) \) measures the likelihood of trajectories forecast by the model \( q_\pi \) under an estimate \( \bar{p} \) of the true predictive distribution. \( H(q_\pi, \bar{p}) \) serves as a measure of the precision of the model. While the two KL divergence metrics address the shortcomings of the mADE \( K \) and mFDE \( K \) metric, they have two limitations. First, while likelihoods allow for comparison of models, their value itself does not have a physical interpretation. Second, the precision metric \( H(q_\pi, \bar{p}) \) requires a model \( \bar{p} \) to estimate the true predictive distribution, making the evaluation metric dependent on the goodness of the estimate \( \bar{p} \).

Thus, to evaluate the precision of the model, we instead measure the percentage of all location co-ordinates predicted by the model that fall on paths in the scene. We manually annotate the scenes in the SDD test set. We assign each pixel to belong to either paths or obstacles. Figure 6a shows an example of the path and obstacle annotations. The % predictions on paths (PoP) metric can be applied to all trajectories generated by the model. It penalizes models that produce a diverse set of random guesses, and its value has a physical interpretation. However, by itself, the PoP metric does not reward multi-modality. Thus, we use it in conjunction with the mADE \( K \) and mFDE \( K \) metrics to evaluate our models. Figure 6b illustrates the significance of using both the mADE \( K \) and PoP metrics.

One caveat with the PoP metric is that for some trajectories in the SDD test set, the ground truth falls on obstacles. For example, these can be cases where agents are walking through corridors underneath buildings but visible through arches, or a small proportion of agents that walk over terrain. 87.88% of the ground truth trajectories fall on our annotated paths, while 12.12% fall on obstacles. For reporting the PoP metric, we only consider the trajectories where the ground truth falls on paths.

4.4 Quantitative Analysis

Comparison with prior approaches: Table 2 shows the mADE \( K \) and mFDE \( K \) values for prior approaches described in section 4.2 and our approach P2T\(_{IRL}\), for the TrajNet split of SDD. Apart from Desire [1] which reports mADE\(_5\) and mFDE\(_5\), all other baselines considered have reported results in terms of mADE\(_{20}\) and mFDE\(_{20}\). To keep evaluation settings consistent with prior work, we also sample 20 trajectories from our model, and report results over a prediction horizon of 4.8 s.

P2T\(_{IRL}\) achieves dramatic improvement in terms of mADE\(_{20}\) and mFDE\(_{20}\) over Social GAN [2]. This can be reasonably expected, since Social GAN does not incorporate static scene context. However, our approach also significantly outperforms the other two GAN based approaches, MATF GAN [10] and Sophie [9], which do incorporate static scene context. Since Desire reports results for \( K=5 \), Table 2 might not represent a fair comparison with P2T\(_{IRL}\). However, we show that P2T\(_{IRL}\) outperforms Desire in terms of mADE\(_3\) (see Figure 7). Finally, we note that P2T\(_{IRL}\) achieves comparable mADE as CF-VAE [12], and slightly outperforms CF-VAE in terms of mFDE. Our approach thus achieves state of the art results on SDD.
The superior performance of P2T$_{IRL}$ and CF-VAE in terms of mADE and mFDE metrics compared to other models, suggests that both models generate a more diverse set of trajectories compared to the GAN and CVAE based approaches. This could be attributed to samples from CFAE’s conditional prior, and plans sampled from our MaxEnt policy, being more informative compared to samples from the simple prior distributions used in the GAN and CVAE based approaches.

Comparison with ablations of our approach: Table 2 shows mADE$_{20}$, mFDE$_{20}$ and PoP values for P2T$_{IRL}$ and the baselines CG, and P2T$_{BC}$. In addition to the standard 4.8s prediction horizon, we also report results for a longer prediction horizon of 10 s.

First, we note that both P2T$_{BC}$ and P2T$_{IRL}$ significantly outperform the CG model in terms of all three metrics, for both prediction horizons considered. This shows the usefulness of the grid based plans. Independent of whether the plans were sampled from a behavior cloning policy, or a MaxEnt IRL policy, trajectories conditioned on grid-based plans are more diverse, and better conform to the scene, compared to trajectories generated by the CG model.

Next, we compare P2T$_{BC}$ and P2T$_{IRL}$. We note that P2T$_{IRL}$ achieves only slightly lower mADE$_{20}$ and mFDE$_{20}$ values compared to P2T$_{BC}$ for both prediction horizons. However, P2T$_{IRL}$ considerably outperforms P2T$_{BC}$ in terms of the PoP metric, especially for the 10s prediction horizon. This suggests that while a diverse set of plans can be sampled from both the BC and MaxEnt policies, the plans sampled from the MaxEnt policy better conform to the scene.

Variation in metrics with number of samples (K): P2T allows for a varying number of trajectories to be sampled from the model. We consider the effect of varying the number of trajectories K sampled from our model, on the mADE$_{K}$ and PoP metrics over a 4.8s prediction horizon.

Figure 7 (left) shows the plots for mADE$_{K}$ vs K for the CG, P2T$_{BC}$ and P2T$_{IRL}$ models. Additionally, we plot mADE values reported for prior approaches. We note that mADE$_{K}$ consistently decreases with K, for the CG, P2T$_{BC}$ and P2T$_{IRL}$ models, which can only happen if the sampled forecasts are diverse. We additionally note that P2T$_{IRL}$ and P2T$_{BC}$ achieve significantly lower mADE$_{K}$ values than the CG model irrespective of K. Additionally, we note that the P2T$_{IRL}$ plot lies below the mADE$_{K}$ values reported for prior approaches for different values of K.

Figure 8 (right) shows the plots for the PoP metric for the CG, P2T$_{BC}$ and P2T$_{IRL}$ models as a function of K. We note that the PoP metric is largely unaffected by the number of trajectories sampled from the models, with P2T$_{IRL}$ outperforming the two baselines across all values of K.
Fig. 9: **Qualitative comparison of forecasts:** Compared to the conditional generative model, the grid based plans lead to trajectories that are diverse, and yet conform to the scene over long prediction horizons. Additionally, our MaxEnt IRL policy leads to better goal driven behavior compared to the behavior cloning policy.
the grid based plans lead to trajectory forecasts that conform to the scene over long prediction horizons, with the MaxEnt policy leading to better scene compliance than the behavior cloning policy.

4.5 Qualitative Analysis

Figure 9 shows example forecasts generated by our model and its ablations. Each column in the figure corresponds to a different prediction instance. The rows (from top to bottom) show (1) the scene and trajectory inputs, (2) trajectories forecast by the CG model over 10s, (3) goal and (4) path SVFs for P2T\textsubscript{BC}, (5) trajectories forecast by P2T\textsubscript{BC} over 10s, (6) goal and (7) path SVFs for P2T\textsubscript{IRL} and (8) trajectories forecast by P2T\textsubscript{IRL} over 10s.

Grid based plans lead to trajectories that conform to the scene. We observe that the trajectories forecast by the CG model (row 2, green) are diverse, but poorly conform to the scene, often running into obstacles like parked cars, buildings or terrain. This suggests that while the mADE\textsubscript{K} loss encourages diversity of forecasts, the conditional generative model fails to generalize to unknown scenes. On the other hand, the trajectories generated by P2T\textsubscript{BC} (row 5, blue) and P2T\textsubscript{IRL} (row 5, red) better conform to the underlying scene. This can be explained by observing the path SVFs for the two models (rows 4 and 7). We note that the states explored by both the BC and MaxEnt IRL policies tend to remain on paths, and avoid obstacles and terrain. We also observe that the models generate trajectories along the states explored by the corresponding policies.

The policies induce a multi-modal distribution over goal and path states. Both the BC and MaxEnt IRL policies are stochastic. Observing the goal and path SVFs of both policies (rows, 3,4,6,7), we observe that the policies induce a multi-modal distribution over the goal and path states. This in turn leads to a diverse set of trajectory forecasts.

The MaxEnt IRL policy is more goal driven than the BC policy. Next we compare the forecasts generated by the P2T\textsubscript{BC} and P2T\textsubscript{IRL} models. We note that goal SVFs of the MaxEnt IRL policy (row 6) are precisely localized, with paths (row 7) leading to these goals. While the BC policy explores paths that conform to the scene(row 3), it is not driven by precisely defined goal states (row 4). This goal driven behavior of our IRL policy can be clearly observed in the second example (column 2), where the IRL policy explores the path leading to the top-left corner of the scene, which the BC policy misses. Additionally, the IRL policy generates paths that lead to parked cars (column 5) and a building entrance (column 6), whereas the BC policy generally explores the scene in the direction of the agent’s motion.

Temporal evolution of forecasts. Figure 10 shows forecasts generated by P2T\textsubscript{IRL} for the same agent at six different instants separated by 1s. We observe that the MaxEnt policy induces a multi-modal distribution over path and goal states, with three prominent modes, leading to locations where paths exit the scene. We observe that as the agent turns upwards at the intersection, the mode leading right becomes less and less prominent, while the modes leading top-right, and top become more prominent. Initially, most of the trajectories sampled from the model lead right, while by the end, most trajectories sampled from the model lead upwards. Thus, we observe that our model conserves all modes of the future distribution, while varying the weights of the modes as more observed context becomes available.
5 CONCLUDING REMARKS

We introduced an approach to forecast trajectories of pedestrians and vehicles in unknown environments conditioned on plans sampled from a grid based MaxEnt IRL policy. We reformulated MaxEnt IRL to learn a policy that can jointly infer goals and paths of agents on a coarse 2-D grid defined over the scene. We showed that our policy infers plausible goals of agents in unknown environments such as points where paths exit the scene, entrances to buildings and parked cars, and paths to these goals that conform to the underlying scene. Additionally, we showed that our policy induces a multi-modal distribution over path and goal states. Next, we introduced an attention based trajectory generator that outputs continuous valued trajectories conditioned on state sequences sampled from our MaxEnt policy. Trajectories sampled from our trajectory generator are diverse and conform to the scene over long prediction horizons, outperforming prior approaches on the TrajNet benchmark split of the Stanford drone dataset in terms of the mADE and mFDE metrics.

REFERENCES

[1] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, “Desire: Distant future prediction in dynamic scenes with interacting agents,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 336–345.

[2] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2255–2264.

[3] N. Deo and M. Trivedi, “Convolutional social pooling for vehicle trajectory prediction,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 1468–1476.

[4] ——, “Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms,” in 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018, pp. 1179–1184.

[5] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, “Multimodal trajectory predictions for autonomous driving using deep convolutional networks,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 2090–2096.

[6] A. Zyner, S. Worrall, and E. Nebot, “Naturalistic driver intention and path prediction using recurrent neural networks,” IEEE Transactions on Intelligent Transportation Systems, 2019.

[7] S. Casas, W. Luo, and R. Urtasun, “Intenetnn: Learning to predict intention from raw sensor data,” in Conference on Robot Learning, 2018, pp. 947–956.

[8] D. Ridel, N. Deo, D. Wolf, and M. Trivedi, “Scene compliant trajectory forecast with agent-centric spatio-temporal grids,” arXiv preprint arXiv:1809.07507, 2019.

[9] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, and S. Savarese, “Sophie: An attentive gan for predicting paths compliant to social and physical constraints,” arXiv preprint arXiv:1806.01482, 2018.

[10] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, “Multi-agent tensor fusion for contextual trajectory prediction,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 12126–12134.

[11] N. Rhinehart, K. M. Kitani, and P. Vernaza, “R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 772–788.

[12] A. Bhattacharyya, M. Hanselmann, M. Fritz, B. Schiele, and C.-N. Straehle, “Conditional flow variational autoencoders for structured sequence prediction,” arXiv preprint arXiv:1908.09008, 2019.

[13] A. Bhattacharyya, B. Schiele, and M. Fritz, “Accurate and diverse sampling of sequences based on a best of many sample objective,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8485–8493.

[14] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, “Precog: Prediction conditioned on goals in visual multi-agent settings,” arXiv preprint arXiv:1905.01296, 2019.

[15] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, “Planning-based prediction for pedestrians,” in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009, pp. 3931–3936.

[16] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, “Activity forecasting,” in European Conference on Computer Vision. Springer, 2012, pp. 201–214.

[17] M. Wulfmeier, D. Z. Wang, and I. Posner, “Watch this: Scalable cost-function learning for path planning in urban environments,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016, pp. 2089–2095.

[18] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, “Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories,” in Conference on Robot Learning, 2018, pp. 894–905.

[19] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in Aaai, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[20] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, “Learning social etiquette: Human trajectory understanding in crowded scenes,” in European conference on computer vision. Springer, 2016, pp. 549–565.

[21] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, “Trajnet: Towards a benchmark for human trajectory prediction,” arXiv preprint, 2018.

[22] M. Wulfmeier, F. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” arXiv preprint arXiv:1507.04888, 2015.

[23] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, “Modeling interaction via the principle of maximum causal entropy,” 2010.

[24] S. Levine, “Reinforcement learning and control as probabilistic inference: Tutorial and review,” arXiv preprint arXiv:1805.00909, 2018.

[25] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[26] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[28] E. Shelhamer, J. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” IEEE Transactions on Pattern Analysis & Machine Intelligence, no. 4, pp. 640–651, 2017.

[29] M. Cramer, “The dgpf-test on digital airborne camera evaluation—a methodology,” in挡住the principle of maximum causal entropy,” 2010.