A. Benchmark Details

A.1. Trajectory Forecasting Details

We validate our method on two forecasting benchmarks, the ETH & UCY benchmark and the Trajnet++ benchmark. The former is a *general* benchmark containing pedestrian trajectories in a variety of scenarios. The latter one is based on a curated meta-dataset that consists of *interacting* scenarios selected from several publicly available subsets, such as WildTrack [13], L-CAS [90] and CFF [4]. This benchmark has been used in a series of recent competitions.

Our evaluation protocol follows the previous work [48, 65, 84]. On the ETH & UCY datasets, we use the leave-one-out approach, where forecasting models are trained on four sub-datasets and tested on the held-out fifth. On the Trajnet++ dataset, we use the official split of the training and test set. One common feature of recent models like Social-STGCNN and Trajectron++ is that the prediction of the primary agent is only conditioned on the states of neighboring agents up to the observation time t_o but not on any steps from t_o to t_p that have already been predicted, *i.e.*, $s^i_{t_p+1} = f(s^i_{1:t_p}, s^{M\setminus i}_{1:t_o})$. While this design choice accelerates training and inference, it makes the forecasting model unaware of the latest states of the nearby agents and causes notoriously high collision rate at long horizon. As such, our evaluation of collision rate for the Social-STGCNN and Trajectron++ is focused on the first four prediction steps where the models still have access to relatively up-to-date information of the surrounding neighbors. Yet, for the Trajnet++ models that perform fully joint prediction in a recurrent manner, $s^i_{t_p+1} = f(s^i_{1:t_p}, s^{M\setminus i}_{1:t_p})$, we measure the collision rate over the entire prediction horizon.

A.2. Reinforcement Learning Details

The original SARL policy [14] requires a linear motion model as well as imitation pre-training to accomplish the reinforcement learning task from sparse reward feedback. These hand-crafted components, however, introduce extra assumptions over the crowd navigation task and make it hard to analyze the sample efficiency of an RL algorithm. To tease apart the effect of our proposed method, we adopt the following dense reward function [85] for the model-free Rainbow algorithm,

$$r(s_t, a_t) = \alpha(d_g^{t-1} - d_g^t) + \begin{cases} -1 & \text{if } d_m^t < 0 \\ 10d_m^t - 1 & \text{else if } d_m^t < 0.1 \\ 1 & \text{else if goal is reached} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where d_g is the Euclidean distance between the robot and its goal, $\alpha = 0.08$ is a control parameter. Other settings are kept the same as Section 4.3.

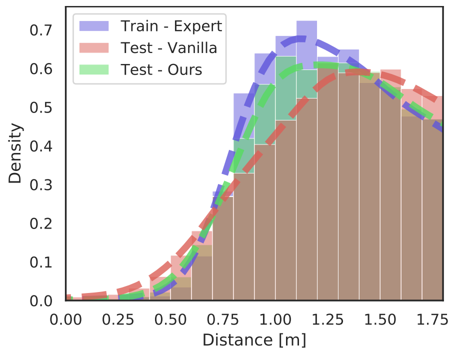


Figure 7: Histogram of the human-robot distance on the crowd navigation task with imitation learning. The vanilla method suffers from the problem of covariate shift in closed-loop sequential predictions, whereas our method results in a much smaller gap between the training and test state distributions.

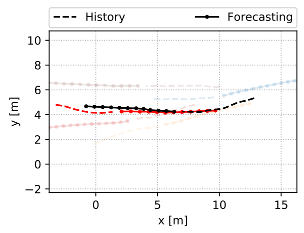
B. Covariate Shift

To further understand the effect of our learning method on closed-loop sequential predictions, we conduct a detailed analysis of the state distribution at test time on the crowd navigation task with imitation learning. Here, we focus on minimum distance between the robot and the other surrounding agents at each frame and collect a set of robot-human distances from 500 test episodes.

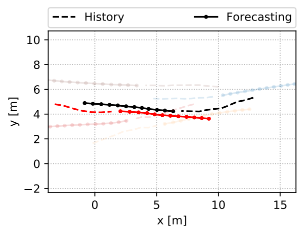
Figure 7 shows the histogram of human-robot distance with different policies. As expected, the density of short-distance states under the expert demonstrator is close to zero, which reflects the expert's high degree of social awareness as well as confirms the lack of dangerous occurrences in the demonstration data. Compared with the training distribution, the test distribution induced by the model from the vanilla imitation learning yields a clear distinction. It exhibits lower density at the distance around 1.0 [m], but higher density in the dangerous regime, *e.g.*, distance smaller than 0.5 [m]. On the contrary, our method results in a test state distribution almost overlapping with the training one over the dangerous states. These results verify that, given the same distribution of the initial state, the model trained with our method visits dangerous states much less frequently and functions in closed-loop operation much more robustly.

C. Qualitative Results

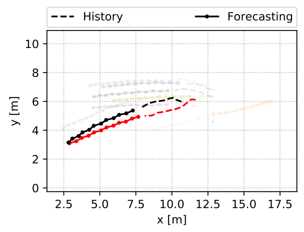
In addition to quantitative comparison, Figure 8 shows the qualitative results of different learning methods in three different interacting scenarios on the Trajnet++ benchmark. The Directional-LSTM [48] trained with the vanilla predictive learning outputs colliding trajectories between the



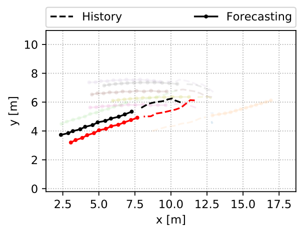
(a) Avoidance - Vanilla



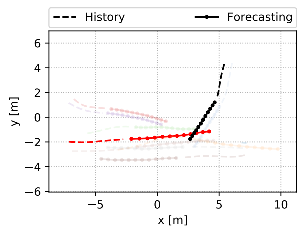
(b) Avoidance - Ours



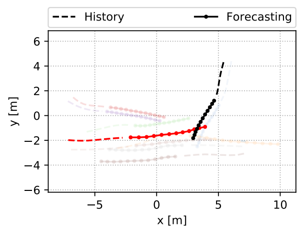
(c) Group - Vanilla



(d) Group - Ours



(e) Other - Vanilla



(f) Other - Ours

Figure 8: Qualitative results of Directional-LSTM [48] models trained with different methods in three *interacting* test cases on the Trajnet++ benchmark [48]. The vanilla method leads to collisions between the primary (black) and the nearby agent (red) at the 4th, 12th, and 10th predicted step on the *Avoidance*, *Group* and *Other* case respectively, whereas our method outputs collision-free trajectories over the whole prediction horizon.

primary agent and its neighbors in these dense scenes. In contrast, our method outputs more socially compliant solutions: in the *Group* scenario, our predicted trajectory for the primary agent stays in the middle of two other neighbors at all time steps instead of sliding towards either of them. In the *Avoidance* scenario, our method adjusts the trajectories of both the primary and the opposite agent cooperatively. Similarly, in the *Other* scenario where pedestrians come from almost orthogonal directions, our method jointly twists the trajectories of these interactive agents, enabling each of them to pass the crowded spot smoothly.