On complementing end-to-end human behavior predictors with planning

Liting Sun, Xiaogang Jia, and Anca D. Dragan
University of California, Berkeley

***Abstract*—High capacity end-to-end approaches for human motion (behavior) prediction have the ability to represent subtle nuances in human behavior, but struggle with robustness to out of distribution inputs and tail events. Planning-based prediction, on the other hand, can reliably output decent-but-not-great predictions: it is much more stable in the face of distribution shift (as we verify in this work), but it has high inductive bias, missing important aspects that drive human decisions, and ignoring cognitive biases that make human behavior suboptimal. In this work, we analyze one family of approaches that strive to get the best of both worlds: use the end-to-end predictor on common cases, but do not rely on it for tail events / out-of-distribution inputs – switch to the planning-based predictor there. We contribute an analysis of different approaches for detecting when to make this switch, using an autonomous driving domain. We find that promising approaches based on ensembling or generative modeling of the training distribution might not be reliable, but that there very simple methods which can perform surprisingly well – including training a classifier to pick up on tell-tale issues in predicted trajectories.**

I. Introduction

Robots that need to share their environments with humans learn predictive models of human behavior, which they use to generate their own behavior in response. Autonomous cars try to predict where other cars will go [7, 21] and what pedestrians will do [17], indoor mobile robots try to predict where the people around them will move [27], and manipulators try to predict how human collaborators will reach for objects in their workspace [18, 15, 6, 10, 14].

When choosing the function class for these learned predictors, high capacity models are very appealing. Recent progress has shown that we can train deep neural networks end-to-end to go from a history of raw state information or even raw sensor data to a distribution over predicted trajectories for a human, implicitly or explicitly extracting relevant features, identifying potential targets in the scene, computing trajectories for each, and assessing their relative likelihoods [2, 16, 26, 24]. Such models dominate the leaderboards in benchmarks for motion prediction (or "forecasting", as it is sometimes referred to) like Argoverse [3] or INTERACTION [25]. They free us from specifying what features might be important or identifying a "theory of mind" for how humans make decisions. Their capacity enables them to represent subtle nuances of human behavior, like people's implicit proxemics preferences, risk aversion level, or anything else that influences where humans go that would be otherwise very challenging to explicitly write down.

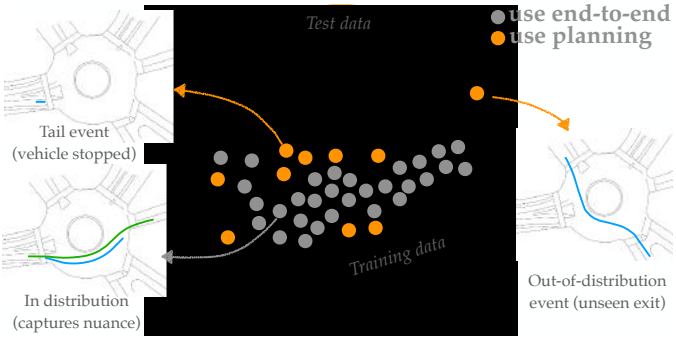

Fig. 1. We analyze methods for using an end-to-end predictor on common cases (gray region), and relying on planning-based prediction outside of that (orange region).

But one challenge that such high capacity, end-to-end models face is their performance in the face of distribution shift or tail events. Our understanding of the nuances of this challenge is still evolving, but there seem to be at least two phenomena at play: one stemming from the model's capacity itself, and one stemming from the way we train these models.

On the capacity side, the function class can represent so many hypotheses that there will exist many of them which fit the data, some based on spurious correlations rather than on the underlying human decision making process that generated those motions. The learner will not be able to disambiguate among them, and can converge to a hypothesis based on a correlate. Over-parametrization increases the ability to represent such hypotheses, lowering average error but possibly increasing error for tail events [20].

Then on the training side, stochastic gradient descent methods introduce their own biases. Sometimes, this bias helps with the capacity issue by pushing the optimization away from the correlates and enabling generalization [22]. However, if there is an easy correlate in the data that helps get most examples right, training might "lazily" converge to that instead of identifying the (more complex) causal variables that explain the data. For instance, a model might learn to predict braking only once the brake light is on, instead of solving the more

complex problem of identifying the need for human to begin braking. This is the more problematic the more information the model has access to, because more correlates exist (the "causal misindentification" [5].

On the other hand, we have *planning*-based predictors. These are based on the idea that human motion results from decisions people make in pursuit of their goals and preferences. Learn their goals and preferences, and leverage planning to generate the corresponding motions [27, 19, 11]. Sticking to the braking example: this type of predictor will easily figure out that a human will need to brake without needing to see their brake lights first. It learns that people want to make progress but avoid collisions and stop at stop signs, and it uses a planner to generate a motion that will do just that. If slowing down is necessary for collision avoidance or because a stop sign is coming up, that is what the planner will do. And indeed, prior work has shown such predictors to be preferable in highly interactive domains, depending on how one collects their training data [4].

However, planning-based predictors suffer from too much inductive bias. They commit to predefined notions (features) for what humans care about which are inevitably both inaccurate (e.g. humans do care about collision avoidance, but might be more sensitive to front collisions than side collisions, for instance), as well as missing important aspects altogether [8]. Further, real people are far from optimal decision makers: we have all sorts of systematic biases, from perception biases to wrong beliefs to risk-aversion to optimism bias and beyond. In short, these predictors are not expressive enough to capture the nuances in human behavior.

Naturally, we would want the best of both worlds. In this work, we analyze one way to strive for that: *use the end-to-end predictor on common cases, but do not rely on it for rare / out-of-distribution inputs – switch to the planning-based predictor there* (Fig. 1). On these inputs, even though the planning-based predictor will not be able to perfectly anticipate human behavior, it will still get the basics right (in our driving domain, for instance, it will output trajectories that stay on the road, avoid collisions, etc.).

We contribute an analysis of different approaches for detecting when to make this switch. We start by outlining natural ways to solve the problem, from detecting out-of-distribution inputs by ensembling and generative methods, to learning to classify when the predictor failed, to using real-time observations of the human's current motion to detect that the predictor is doing a poor job at anticipating what is happening. In order to assess and compare their ability to make more accurate predictions on difficult inputs by switching to the planning-based predictor when they have to, but keeping the end-to-end one when it performs well, we create tests sets that purposefully introduce domain shift. We measure each method's ability to accurately identify the shift, as well the resulting "hybrid" predictor's accuracy.

Our findings first support this hypothesized relationship between end-to-end predictors and planning-based ones. Our end-to-end predictor does much better in-distribution (on a validation set drawn from the same distribution as the training data) than the planning-based one. On the other hand, it is not robust to the shifts and perturbations we introduce, whereas the planning-based one stays remarkably consistent. One of our contributions is merely showcasing this in the driving domain.

As for the switching methods, the results are quite interesting. Training an ensemble and using disagreement as a stand in for "the predictor is uncertain here" [13] fails to identify many of the cases that it should, because the members agree even when making the wrong prediction. Switching based on observing the real human online to take actions that the predictor is assigning low probability to is very reliable, but it does introduce a significant switching delay because the robot has to observe enough such "low probability" human actions. Surprisingly, a simple classifier that we train to label predictions as good or bad based on the predictor's performance on training data is also very reliable. The classifier is not meant to quantify uncertainty or detect out of distribution issues, but it implicitly does that by learning to pick up on features of predicted trajectories that are suggestive of something having gone wrong, like going off the road or not stopping at a stop sign.

Overall, if we can combine the power of end-to-end prediction with the robustness we get out of planning-based prediction, robots that act around people will be able to anticipate and adapt to nuanced human behavior while still maintaining reasonable performance in the long tail. Our work analyzes one family of approaches that strive for this combination, with somewhat surprising but promising results. We intend this as a first step in this direction, starting a discussion into what approaches and ideas are most promising when it comes to this general goal of getting the best out of both worlds.

II. Methods

A. Problem Statement

We are given a training set $\mathcal{D}$ consisting of tuples (x, h, ξ), where x is the state of the environment (including map data), h is the history of motion observation for all agents in the scene, and ξ is the trajectory label for the target agent. We are also given two predictors trained on (a subset of) this data: 1) a high capacity "end-to-end" model f_{e2e} that learns to map (x, h) to ξ; 2) a planning-based model f_{plan} that learns a cost function that explains the motions observed in training, and optimizes it to generate predictions ξ for the target agent (and the other agents in the scene).[1]

[1]Note that planning-based models need to make explicit joint or iterative predictions about other agents as well, since the target agent optimizes to avoid collisions with them.

The goal is to output a switching detector $\sigma : (x, h, f_{e2e}(x,h)) \mapsto \{0,1\}$ that determines, based on a new input (x,h) from a test distribution $\mathcal{T}$ and optimally the prediction $\hat{\xi} = f_{e2e}(x,h)$ on that new input, whether this is an input on which the end-to-end predictor will have high error. Needless to say, σ does not get access to the test distribution $\mathcal{T}$.

Armed with σ, the robot, upon encountering a new input (x,h), can predict an agent's trajectory using

$$f_\sigma(x,h) = \begin{cases} f_{e2e}(x,h), & \sigma(x,h,f_{e2e}(x,h)) = 0 \\ f_{plan}(x,h), & \sigma(x,h,f_{e2e}(x,h)) = 1 \end{cases}$$

We discuss below four natural ways to train such a σ.

Aside: Note that this makes the assumption that the end-to-end model will only fail in rare or out-of-distribution situations, where our best bet will be the planning model. Depending on how these are trained, this will not always be true (end to end models might break on common cases as well, and planning-based predictors might not be better on rare cases). Nonetheless, we choose to focus on this setting because we believe it to be most representative of how these predictors will be developed in the real world, where end to end models will have high enough capacity to fit the average case well, and planning-based models will be of low enough capacity to generalize well. Our experiments below support these assumptions.

B. Preliminaries: Example Predictors

We use the INTERACTION dataset [25] to train our predictors. We use segments of the data of 40 timesteps from the beginning, middle, and end of runs (10 timesteps as the history, and the other 30 timesteps as the label). We train a LSTM model for our end-to-end predictor with pooling layers to encourage the interactions among agents in a similar way as social-LSTM [1] (see Appendix B for details), and we use Inverse Reinforcement Learning (IRL) to recover a cost function for our planning-based predictor using features for collision avoidance, progress, lane keeping, etc., similar to [12, 23]. To enable the model to predict collision avoidance with other agents, we use the cost function to first optimize predictions for further away agents, and iteratively compute trajectories for nearby agents until we reach the target agent. To avoid counfounding effects from the need to predict the geometric intent of agents, we provide both predictors with the overall reference path the agent is following (e.g. which exit they are taking in a roundabout). Further details are in Appendix A.

C. Ensemble Disagreement

Following [13], we note that if we train an ensemble instead of a single end-to-end predictor, ensemble disagreement can be used as a measure of uncertainty or confidence. When the members of the ensemble make contradicting predictions, this is a signal that we are out-of-distribution.

We select 5 different models as the ensemble members. All the members share the same structure as the end-to-end predictor, and they all train with the same training data. However, each of them is trained with different initialization, and the order of examples they use naturally differ because of SGD (stochastic gradient descent). To quantify their disagreement, we take the most-probable prediction from each of them, denoted by $\hat{\xi}_j (j = 1, 2, \cdots, 5)$, and calculate the variance on their final positions. Namely, suppose the final positions on $\hat{\xi}_j$ is (x_j^F, y_j^F), then the metric for the disagreement is given by[2]

$$E_{disagree} = \max\{var\{x^F_{j=1,\cdots,5}\}, var\{y^F_{j=1,\cdots,5}\}\}$$

$$\sigma(x,h) = \begin{cases} 1, & E_{disagree} \geq \tau \\ 0, & otherwise \end{cases}$$

with τ a threshold we tune on the training set.

D. Generative Modeling of the Training Distribution

Another way to detect whether a new input is in the training distribution is to train a generative model of the inputs – train a GAN (generative adversarial network) on inputs (x,h), and use its discriminator at test time as our σ to tell whether the test input is "real", i.e. from the training data, or not.

We use the architecture from Fig. 2. In our experiments we only focus on the h side of the input (the history) and not the whole scene x because of the difficulty in generating scene configurations, but in principle that would be possible as well. The generated data is labeled as "fake", the training data is labeled as "real".

E. Classifying Poor Predictions

The third approach is to simply train a classifier to explicitly distinguish whether a prediction is good enough. That is, instead of attempting to detect whether an input is out of distribution or rare, simply look at the prediction and classify whether it's correct – with the hope that discriminating that a prediction is poor is an easier task for a neural network that producing a good prediction in the first place. The classifier tries to learn a function $\sigma : (x, h, f_{e2e}(x,h)) \mapsto \{0,1\}$ from training data $\mathcal{D}_{\text{classifier}}$ that we auto-label based on the average distance error (ADE) between the predicted trajectory and the ground-truth trajectory in the predictor training data. We label all predicted trajectories that generate

[2]We experimented with several distance metrics in our experiment and this performed the best, though the results are very sensitive to the choice of a metric, so we encourage practitioners to both analyze different metrics when trying this, as well as different ways of creating diversity in the ensemble – ours is but a starting point, so we used the simplest approach.

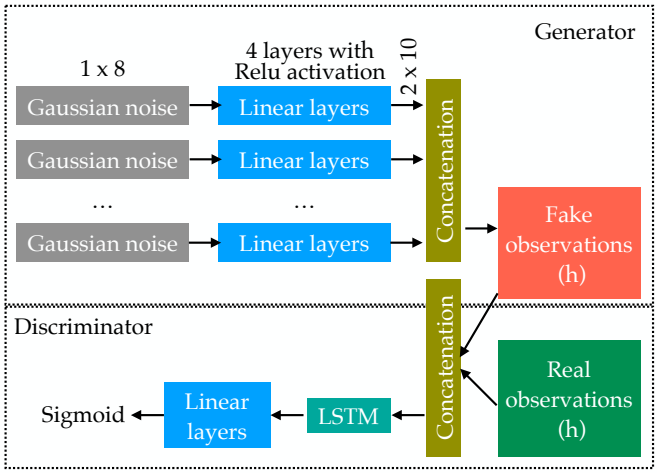

Fig. 2. The structure for the generative model

large ADEs (2 sigmas beyond the mean ADE of the training set) as bad predictions (1).

Fig. 3 shows the structure of the classifier, similar to that in the discriminator of the generative modelling approach (but with access to the predicted trajectory as well). We use a softmax loss for training.

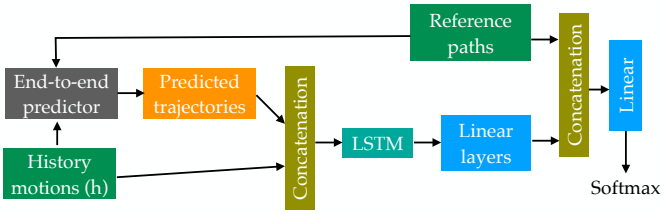

Fig. 3. The structure for the classifier

F. Online (Bayesian) Failure Estimation

The fourth approach we investigate is a bit different in what it has access to. Rather than training something ahead of time, here the robot at test time receives ground truth human observations from the human it is trying to predict, and uses them to determine whether its predictor is operating accurately or not in that particular new setting. This is directly inspired by [8]: if at test time the human takes actions that are too low of a probability under the predictor, we conclude that the predictor is not correctly handling the current situation.

Suppose that at time t, we have the predicted trajectory as $\hat{\xi}_t = f_{e2e}(x,h)$. After another m timesteps (m is smaller than the length of the predicted trajectory), we have new human observations, denoted as $\xi_{t:t+m}$. From the discrepancy between $\xi_{t:t+m}$ and $\hat{\xi}_t$, we can infer whether the prediction $\hat{\xi}_t$ is good or not. An illustrative example is given in Fig. 4.

In general, for probabilistic predictors where $f_{e2e}(x,h)$ is a probability distribution, we can use

$$\sigma(\xi_{t:t+m}, f_{e2e}(x,h)) = \begin{cases} 1, & P(\xi_{t:t+m}|f_{e2e}(x,h)) < \tau \\ 0, & otherwise \end{cases}$$

with τ a threshold tuned based on the training data. For predictors that only output trajectories, we can use a proxy distribution based on distance, $P(\xi|\hat{\xi}_t) \propto \exp\{-\frac{1}{m} L_2(\hat{\xi}_{t:t+m}, \xi_{t:t+m})\}$.

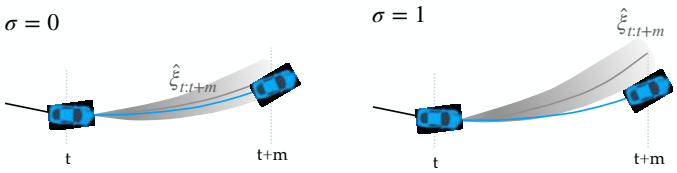

Fig. 4. An illustrative example of the Bayesian failure estimation approach: gray bands are two different predicted distributions at time t and the blue one is the new observed trajectory at time $t+m$.

III. Experiments

To analyze how promising these switching approaches are for detecting that we should plug in the planning-based predictor instead of using the default end-to-end one, we need data where tail events and distribution shift occur. While tail events will happen for most training distributions we would encounter, distribution shift is something we decided to purposefully introduce in a controlled way. We use a real driving data set (the INTERACTION data [25]), and design three experiments that use different train and test sets to probe at different types of shift – from introducing noise to the input, to testing on a new exit from a roundabout where we didn't have data (new reference paths), all the way to switching to an entirely new map.

Note these shifts sound more drastic than they are. We chose them not because autonomous cars might navigate new maps – since the predictors only focus on 30-step snippets of the overall trajectory and have access to the geometric reference that the vehicle is following, this is more akin to changing the configurations of the road and other agents than putting the robot in an entirely new region. But as experiment designers, it frees us from having to slice the data by what are common configurations vs. new ones – if we had a metric for this, that would be our switching method in the first place!

A. Experiment Design

Independent Variables. We manipulate which switching method σ is used, from our four methods in section II, and adding always 0 (use f_{e2e} only) and always 1 (use f_{plan} only). We also manipulate whether shift is present or not, and the type of shift: new reference paths (new exit), new map, and noise.

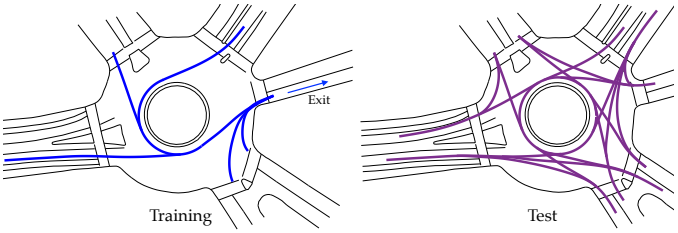

Fig. 5. The training and test domains for experiment I, where we introduce new exits. The predictors only deal with local 30-step snippets and have access to the ground truth reference path.

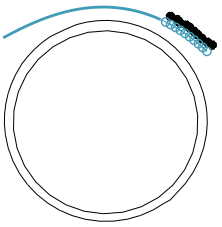

Fig. 6. The training and testing domain for experiment III, where we introduce noise (blue circle: real observations; black star: observations with added noise; blue curve: ground truth future trajectories).

In Experiment I, as shown in Fig. 5, all the training data includes trajectories that exit only through one selected lane (left), but in the test domain, trajectories following all possible reference paths are included (right). In the training domain, 1948 examples were included.

In Experiment II, we train on the map for Experiment I (all exits, 39764 examples), but test on different maps (still providing the reference as an input).

In Experiment III, we train on all maps, and add Gaussian noise to the history of observations at test time. We set the Gaussian parameters as $\mu = 0.5, \sigma = 0.1$, and the effect this has is visualized in Fig. 6.

The switching methods have access to the same training data as the predictors. For each experiment, we also have a validation set from the same distribution on the training data, and evaluate the methods on both validation and test.

Metrics. We have two measures: accuracy and performance. Accuracy refers to the switching method's ability to predict whether the end-to-end predictor will have high error. Performance refers to the resulting hybrid predictor (which uses planning-based when $\sigma = 1$) error, measured as average distance error (ADE) between the predicted trajectories and the ground-truth ones, which is standard in motion prediction [1, 2, 26, 24, 16].

Hypotheses. We hypothesize that H1) f_{e2e} has lower validation set ADE than f_{plan}, but higher test set ADE; and H2) f_σ (the hybrid predictor) has lower ADE than both in both validation and test by using f_{plan} on tail events and out of distribution inputs where f_{e2e} struggles. However, the goal of the analysis is to compare the different approaches for σ and establish their potential strengths and weaknesses.

	Val acc.	Test acc.	Val ADE (m)	Test ADE (m)
f_{e2e} only (LSTM)			0.527	2.9648
f_{plan} only (IRL)			0.721	0.80
ensemble	81.78%	83.05%	0.5464	1.2699
GAN	75.13%	84.53%	0.5191	0.9118
classifier	**85**%	85%	0.5377	**0.6804**
30-step online Bayesian	**100**%	**100**%	**0.4177**	**0.6162**
5-step online Bayesian	68.3%	**89**%	**0.4892**	0.8235

TABLE I
RESULTS OF EXPERIMENT I

B. Experiment I - Generalizability across different references

Performance of the predictors. Fig. 7 shows the performance of the two predictors in the validation and test sets. In validation, f_{e2e} (LSTM) has lower error than f_{plan} (IRL) – .527 vs .721 (see Table I). There are some tail events with high error though. On test, the opposite is true. This is in line with H1. Note that f_{plan} has relatively steady performance in the face of the shift, whereas f_{e2e} goes from much better to drastically worse.

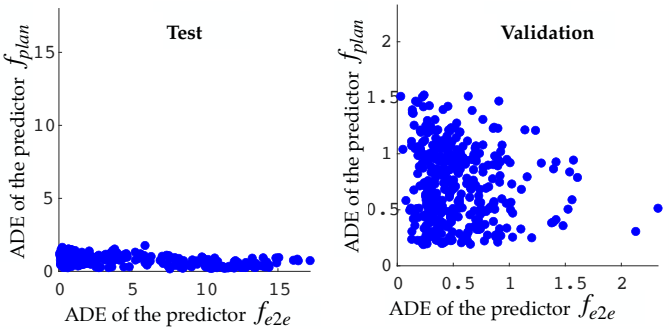

Fig. 7. The performance of the two predictors in validation and test domains in Experiment I. Note the difference in scale.

Failure modes of the predictors. *Training domain:* Many of the f_{e2e} failures were in cases where the vehicle is stopped or moving slowly due to stop signs or traffic congestion. These are relatively rare events. Fig. 8 shows two such cases where the predictor assumes progress when the ground truth stays put.

f_{plan} has no issue predicting the stop because it models people as following rules and avoiding collisions. However, it struggles on cases where its structure is wrong. Unlike f_{e2e}, it uses the reference path as a hard constraint, and some of our data has the wrong reference (Fig. 9, left). This is akin to what might happen in the real world where the map annotations are wrong, and a planning-based predictor will stick to them. It also sometimes creates unnatural motion (Fig. 9, right)

because it is missing important aspects of what people want or converging to bad local optima.

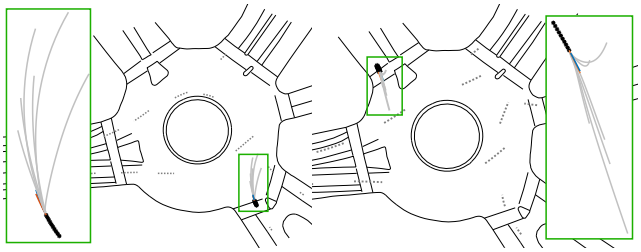

Fig. 8. A failure mode of f_{e2e} on the training domain, where the predictions (gray) make progress when the ground truth (blue, see the zoomed-in details) almost stays put. The black dotted lines in the center are observations for surrounding vehicles.

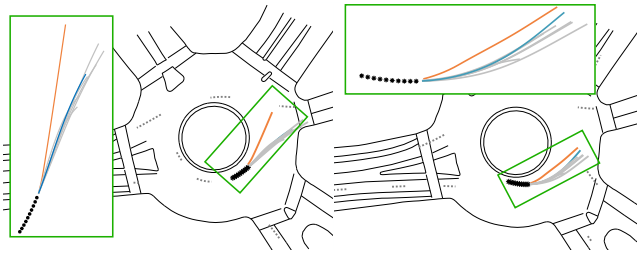

Fig. 9. Failure modes of f_{plan} on the training domain, where it has the wrong reference path (left) or missing features / converging to poor optima (right). The f_{plan} predictions are in orange, f_{e2e} in gray, ground truth in blue (similar to f_{plan}).

Testing domain: f_{e2e} fails at test time either by failing to pursue the correct reference (e.g. predicting drastic turns to go to the exit it was trained on), or by ignoring road geometry and moving through obstacles. Both are likely to be due to the shift in distribution. Fig. 10 shows some examples.

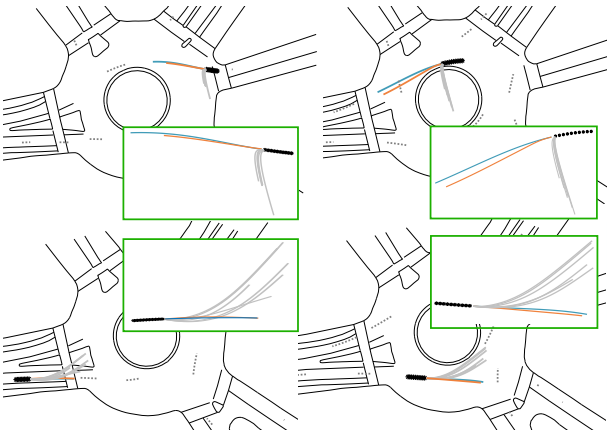

Fig. 10. Failures of f_{e2e} in the test domain for Experiment I. The predictions in gray aggressively pursue the wrong exit and/or ignore road obstacles. f_{plan} predictions are in orange, similar to ground truth (blue).

Performance of switching approaches. Fig. 11 - Fig. 14 show the four approaches ability to pick up on high error for f_{e2e} in validation and test. Right off the bat, we see that the ensemble struggles – sometimes it agrees on data it shouldn't (high error on x axis), and disagrees on data it should agree on (low error on x axis). It does catch the biggest outliers in validation, but fails to catch some high error points in test. This could be explained by the bias in SGD – despite the fact that there are many hypotheses explaining the data that the class can represent, and ideally the ensemble members would converge to different ones, in reality the bias in SGD itself might lead to converging to similar hypotheses.

The GAN has mixed performance, though better. Many of the test cases look similar enough to the training data that the discriminator misses them.

On the other hand, the classifier works remarkably well (Fig. 13, picking up on many issues in validation and test, with some false positives. This is interesting, because the classifier is only trained in-domain, so how can it give the correct label off-distribution? The answer is that there are enough failures in the training domain (on those tail events) that the classifier learns the "tell-tale signals" that the prediction has gone wrong. Instead of focusing on the input domain itself (like the GAN), it focuses on the prediction, and picks on patterns like "whenever the prediction goes off the road, we get high error", or "whenever there is a stop sign and the prediction is not stopping, we get high error". While this will not catch more subtle wrong predictions, it seems to be an effecting way of automatically learning metrics for sanity checking predictions (without having to think ahead of time of what these metrics need to be and craft them by hand).

When given enough time steps of observation (Fig. 14), the online failure detector works perfectly. Unfortunately, this is not very practical because it causes a long delay between the predictions being poor and switching to f_{plan}. When we try to make the call with only 5 steps (Fig. 15), results are much worse.

Table I shows what effects this accuracy end up having for the f_σ error. On validation, all methods maintain the good performance of f_{e2e}, with the online failure detector improving it. We see here a discrepancy between accuracy and error, where even if the 5-step online detector has lower accuracy than other methods, it ends up better error. This is perhaps because the yes/no metric does not capture the magnitude of the error. On the test set, all methods drastically improve performance, with the classifier and online failure detector performing the best.

C. Experiment II - Generalizability across different maps

Performance of the predictors. The performance of the two predictors matches Exp I: f_{plan} is stable, f_{e2e} is much better in validation and much worse in test, as shown in Table II, as well as in Fig. 20 in the Appendix C.

Failure modes of the predictors. We see that with this testing domain, f_{e2e} sometimes fails to follow the

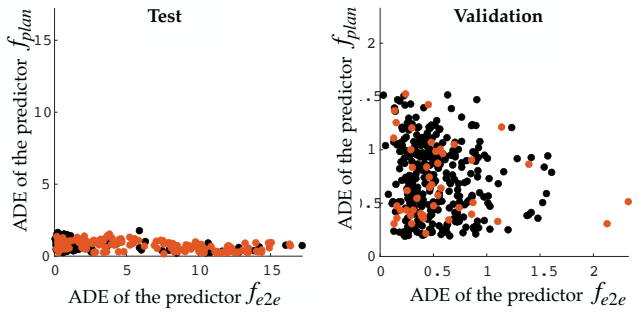

Fig. 11. The ADE scatter with ensembling as a performance detector (black: good cases for the lstm model; red: good cases for the irl model)

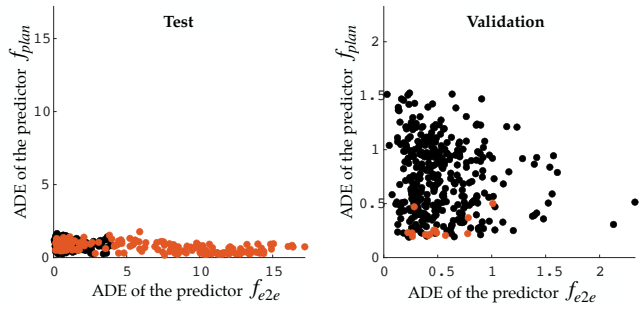

Fig. 12. The ADE scatter with GAN as a scenario detector (black: good cases for the lstm model; red: good cases for the irl model)

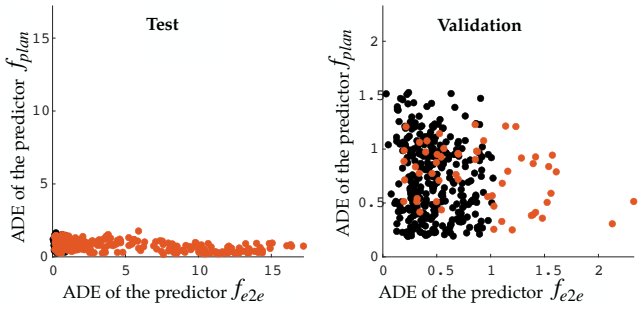

Fig. 13. The ADE scatter with a trained classification model as a performance detector (black: good cases for the lstm model; red: good cases for the irl model)

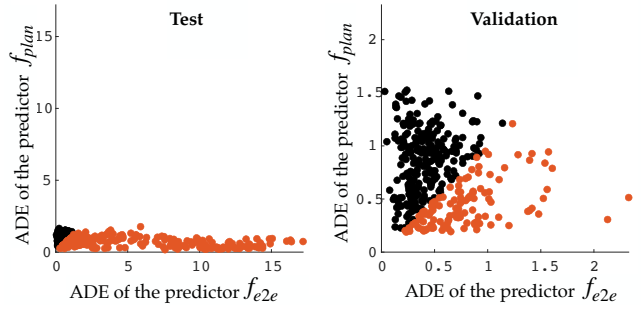

Fig. 14. The ADE scatter with online bayesian based on 30-step trajectories (black: good cases for the lstm model; red: good cases for the irl model)

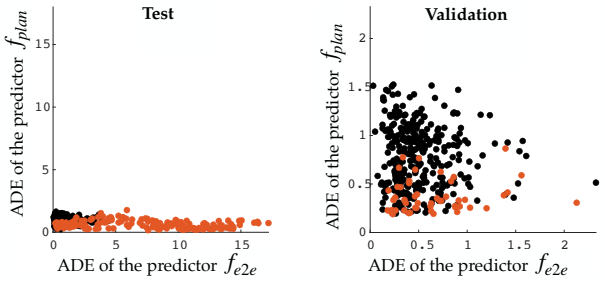

Fig. 15. The ADE scatter with online bayesian based on 5-step trajectories (black: good cases for the lstm model; red: good cases for the irl model)

	Val acc.	Test acc.	Val ADE (m)	Test ADE (m)
f_{e2e} only (LSTM)			0.367	1.691
f_{plan} only (IRL)			0.8224	1.0634
ensemble	80.3%	82.4%	0.3888	0.9369
classifier	**93**%	**93**%	0.372	**0.742**
30-step online Bayesian	**100**%	**100**	**0.3280**	**0.6435**
5-step online Bayesian	52.3%	85%	**0.3475**	0.8461

TABLE II
RESULTS OF EXPERIMENT II

reference, or produces trajectories that stray off the road (Fig. 16), while f_{plan}'s structure enables it to seamlessly produce reasonable trajectories despite the difference in domain. The Appendix C provides further examples in Fig. 25 from another test map.

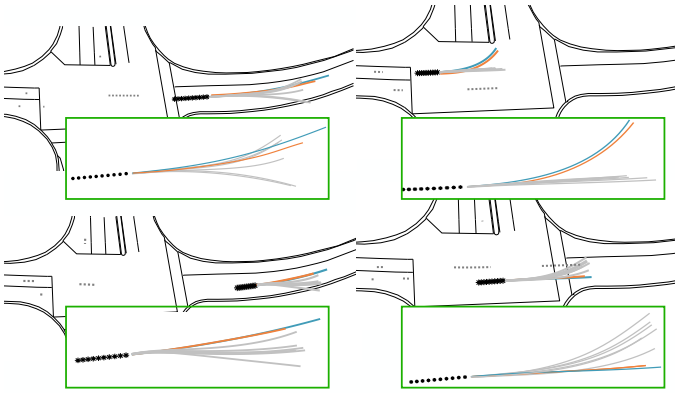

Fig. 16. Failures of f_{e2e} when tested on a new map. Its predictions in gray, f_{plan} predictions in orange, and ground truth in blue.

Performance of switching approaches. Table II summarizes the findings, mirroring Exp I.

D. Experiment III - Robustness to added noise

Performance of the predictors. The performance of the two predictors is analogous, although a bit more extreme than in the previous two experiments (see Table III): here, the end-to-end predictor really struggles under Gaussian noise so its test domain error is drastically higher. f_{plan}

	Val acc.	Test acc.	Val ADE (m)	Test ADE (m)
f_{e2e} only (LSTM)			0.3561	7.0858
f_{plan} only (IRL)			0.8194	**1.0434**
ensemble	79.24%	82%	0.3900	2.2693
GAN hybrid	71.68%	87.44%	0.3561	1.4649
classifier	**92.8**%	**99.21**%	0.349	1.0588
30-step online Bayesian	**100**%	**100**%	**0.2946**	**0.9016**
5-step online Bayesian	75.34%	94.61%	**0.3192**	1.5211

TABLE III
RESULTS OF EXPERIMENT III

remains relatively stable, but takes a bit of a hit as well compared previous settings.

Failure modes of the predictors. Fig. 17 shows some failure cases due to the added noise for the f_{e2e} predictor (qualitatively similar to the failures we saw in the previous experiments – not following the reference and/or going off the road). f_{plan} is also affected by the noise via wrong speed and orientation estimation for the vehicle, leading to less drastic errors but e.g. going much slower than the ground truth – see Fig. 18.

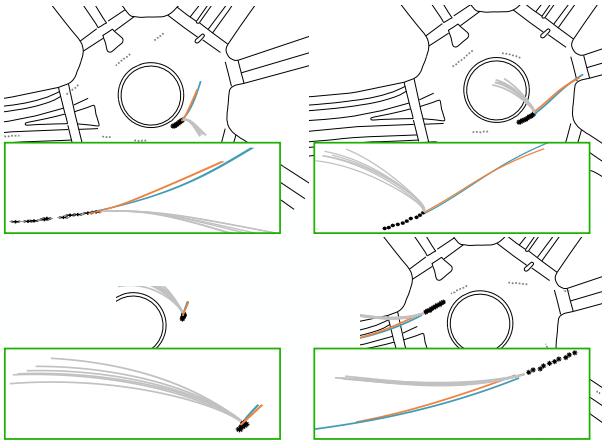

Fig. 17. Failures of f_{e2e} due to noise in the input (predictions in gray, compared to ground truth in blue and f_{plan} in orange).

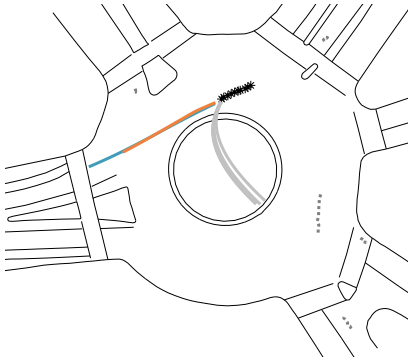

Fig. 18. A failure of f_{plan} due to noise in the input (predictions in orange, ground truth in blue speeds up more).

Performance of switching approaches. Table III shows the accuracy and resulting error for the different σ approaches. Again, these reproduce what we see before in Experiments I and II. The Appendix C contains the analogous scatter plots for the error.

IV. Discussion

Summary of idea and approaches. We put forward the hypothesis that high capacity end-to-end predictors will struggle with tail events and distribution shift, while planning-based predictors will be more robust but will not perform as well on common cases. We therefore want to get the best of both worlds, and we investigate one way to attempt this: detect when we're in one of those non-common cases, and switch to the planning-based method. We study natural approaches for this switch: 1) detecting that we are in a new area because multiple hypotheses that were consistent with training are disagreeing ("ensemble"), 2) discriminating that we're not in the training distribution by training a generative model of our data ("GAN"), 3) classifying whether the end-to-end model has high error by looking at the prediction it is outputting ("classifier"), and 4) detecting online that the model keep assigning low probability to what the current human is actually doing ("online Bayesian failure detector").

Summary of findings. Our first empirical contribution is to test our hypothesis above. By purposefully inducing distribution shift, we show just how robust planning-based predictors can be. This is important in and of itself, because in the era of end-to-end high capacity models we tend to forget the strengths of these high inductive bias methods – it is intuitive that this robustness property would hold, but we quantify it in 3 experiments, and find in 2 of them a quite remarkable stability despite strong shift in the distribution. To the best of our knowledge, such a comparison between end-to-end models and planning-based approaches had not been performed.

Our second empirical contribution is a first analysis of these switching methods, which gives us some intuition about what to expect as they get further developed. We see that ensembles, while promising in theory, might not by default disagree when they should, perhaps because of the bias in SGD-like optimizers to converge to somewhat similar hypotheses despite stochasticity in initialization and sampling data. Also, we found that the ensembling performance relies on the metrics that we use. In contrast, we see that a classifier based on the predictor's performance in training data, which fires when the predictor is wrong, might actually be more powerful than it sounds: while it does not build a notion of being "out of distribution", it can learn to pick up on and recognize eggregious prediction mistakes (like going off-road). Of course, predictions that are subtly wrong (i.e. plausible) could escape such a method. It also seems like an online detector that figures out the predictor is

mis-behaving and keeps attributing low probability to the human's actual actions is by far the most reliable – of course, this would come at a delay, so it should be seen as a must-have fallback switching mechanism (our findings suggest everyone should use it, but also have other ways to detect a switch is needed that have less delay).

Limitations and future work. Our work is limited in many ways. First, we do not study the prediction's impact on the robot's planning or behavior generation – just improving prediction accuracy does not necessarily lead to robot behavior improvements (unless the robot itself works via imitation learning and we think of this method in that domain instead of the human modeling domain). Second, we studied very basic predictors (for both end to end as well as planning) and eliminated the need for intent prediction. Note however that the goal of our work is not to improve upon the latest state of the art predictors – the point of our paper is that one can always improve the end to end learners with better architectures, data augmentation, better training, but the issues of domain shift and tail events are unlikely to go away. Finally, we operated under the hypothesis that we want to use the end-to-end predictor primarily (and indeed, in our results it superior to the planning-based approach for the common cases), but switching techniques based on treating the two predictors as experts and obtaining a mixture are also possible.

Going further in this research, we are excited to pursue the switch idea as a data augmentation mechanism: switch to the planning-based predictor, but use the data generated this way to augment the real training data and therefore improve the robustness of the end-to-end model.

Acknowledgments

We thank the members of the InterACT lab for helpful discussion and ideas, especially Micah Carroll and Rohin Shah whose "fill-in-the-blanks" work was inspiration for this project. This work was partially supported by NSF CAREER and AFOSR.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[2] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*, pages 86–99. PMLR, 2020.

[3] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[4] Rohan Choudhury, Gokul Swamy, Dylan Hadfield-Menell, and Anca D Dragan. On the utility of model learning in hri. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 317–325. IEEE, 2019.

[5] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *arXiv preprint arXiv:1905.11979*, 2019.

[6] Hao Ding, Gunther Reißig, Kurniawan Wijaya, Dino Bortot, Klaus Bengler, and Olaf Stursberg. Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction. In *2011 IEEE International Conference on Robotics and Automation*, pages 5875–5880. IEEE, 2011.

[7] Katherine Driggs-Campbell, Roy Dong, and Ruzena Bajcsy. Robust, informative human-in-the-loop predictions via empirical reachable sets. *IEEE Transactions on Intelligent Vehicles*, 3(3):300–309, 2018.

[8] Jaime F Fisac, Andrea Bajcsy, Sylvia L Herbert, David Fridovich-Keil, Steven Wang, Claire J Tomlin, and Anca D Dragan. Probabilistically safe robot planning with confidence-based human predictions. *arXiv preprint arXiv:1806.00109*, 2018.

[9] Arne Kesting, Martin Treiber, and Dirk Helbing. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605, 2010.

[10] H. S. Koppula and A. Saxena. Anticipating human activities for reactive robotic response. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2071–2071, 2013.

[11] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

[12] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646. IEEE, 2015.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6405–6416, 2017.

[14] Przemyslaw A Lasota and Julie A Shah. Analyzing the effects of human-aware motion planning on

close-proximity human–robot collaboration. *Human factors*, 57(1):21–33, 2015.

[15] Przemyslaw A Lasota and Julie A Shah. A multiple-predictor approach to human motion prediction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2300–2307. IEEE, 2017.

[16] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020.

[17] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–782, 2017.

[18] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 299–306. IEEE, 2013.

[19] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, volume 2. Ann Arbor, MI, USA, 2016.

[20] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pages 8346–8356. PMLR, 2020.

[21] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3399–3406. IEEE, 2018.

[22] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[23] Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2111–2117. IEEE, 2018.

[24] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021.

[25] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[26] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*. PMLR, 2020.

[27] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, 2009.

V. Appendix

A. The planning-based predictor

1) Learning the cost functions: We assume that the cost function of human drivers is a linear combination of a set of predefined features. Thus, given a tuple (x, h, ξ) in the training set D, the cost function associated with it can be described as $C(\xi, \hat{\xi}_O; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{f}(\xi, \hat{\xi}_O)$. Note that $\hat{\xi}_O$ represents the estimated trajectories of all other surrounding agents, and $\mathbf{f}$ is the feature vector and $\boldsymbol{\theta}$ represents driver's preference over different elements in $\mathbf{f}$. With that, based on the principle of maximum entropy, we have

$$P(\xi|\hat{\xi}_O; \boldsymbol{\theta}) \propto \exp\{-\beta C(\xi, \hat{\xi}_O; \boldsymbol{\theta})\}, \tag{1}$$

where β is a hyper-parameter that controls to what levels the human behaves as a rational optimizer. Hence, the log-likelihood of the training set D (with N tuples) can be given by

$$\log P(D|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \frac{\exp\{-\beta C(\xi_i, \hat{\xi}_{O,i}; \boldsymbol{\theta})\}}{\int \exp\{-\beta C(\tilde{\xi}, \hat{\xi}_O; \boldsymbol{\theta})\} d\tilde{\xi}}. \tag{2}$$

By maximizing the log-likelihood, we can find the optimal parameter $\boldsymbol{\theta}^*$ that represents humans' preferences in real driving.

2) The feature set: The features we selected to parametrize the trajectories in the planner-based predictors can be grouped as follows:

- Speed - The incentive of the human driver to reach a certain speed limit v_{lim} is captured by the feature

$$f_v(\xi) = \sum_{t=0}^{L} (v_t - v_{\text{lim}})^2 \tag{3}$$

  v_t is the speed at time t along trajectory $\hat{\xi}$ and L is the length of the trajectory in time.
- Traffic - In dense traffic environment, human drivers tend to follow the traffic. Hence, we introduce a feature based on the intelligent driver model (IDM) [9]

$$f_{\text{IDM}}(\xi) = \sum_{t=0}^{L} (s_t - s_t^{\text{IDM}})^2 \tag{4}$$