RuleFuser: Injecting Rules in Evidential Networks for Robust Out-of-Distribution Trajectory Prediction

Jay Patrikar, Sushant Veer*, Apoorva Sharma*, Marco Pavone and Sebastian Scherer

***Abstract*—Modern neural trajectory predictors in autonomous driving are developed using imitation learning (IL) from driving logs. Although IL benefits from its ability to glean nuanced and multi-modal human driving behaviors from large datasets, the resulting predictors often struggle with out-of-distribution (OOD) scenarios and with traffic rule compliance. On the other hand, classical rule-based predictors, by design, can predict traffic rule satisfying behaviors while being robust to OOD scenarios, but these predictors fail to capture nuances in agent-to-agent interactions and human drivers' intent. In this paper, we present RuleFuser, a posterior-net inspired evidential framework that combines neural predictors with classical rule-based predictors to draw on the complementary benefits of both, thereby striking a balance between performance and traffic rule compliance. The efficacy of our approach is demonstrated on the real-world nuPlan dataset where RuleFuser leverages the higher performance of the neural predictor in in-distribution (ID) scenarios and the higher safety offered by the rule-based predictor in OOD scenarios.**

I. Introduction

Autonomous vehicles are increasingly venturing into complex scenarios that are common in dense urban traffic. Safely navigating these scenarios while interacting with heterogeneous agents like drivers, pedestrians, cyclists, etc. requires a sophisticated understanding of traffic rules and their impact on the behavior of these agents. However, most modern trajectory predictors are developed by merely imitating trajectories from driving logs with no direct supervision on traffic rules. In fact, direct supervision of traffic rules is not plausible as the training data would require examples of both traffic rule satisfaction and violation. Furthermore, the performance of these predictors deteriorates in out-of-distribution (OOD) scenarios. On the other hand, rule-based predictors account for traffic rules and are more robust to distribution shifts, but they struggle with nuanced human driving behavior (which can often violate strict traffic rules) and multi-modal intents. In this paper, we develop a neural prediction framework, RuleFuser, that combines the benefits of both the learning-based and rule-based predictors.

RuleFuser is inspired by the observation that learning-based predictors can often outperform rule-based predictors in performance metrics, but can behave much more erratically (e.g. predicting that vehicles may collide or go off road) than rule-based predictors, especially on out-of-distribution (OOD) data. RuleFuser leverages this observation through an evidential Bayes approach inspired by PosteriorNet [1]. A rules-based predictor designed to comply with traffic rules provides an informative prior over possible future trajectories the agent may take, which a learned neural network model subsequently updates to yield a data-driven posterior distribution over future trajectories. Importantly, the strength of this update is controlled by an estimate of the "evidence" the training dataset provides for this example. In this way, the powerful learning-based model is trusted on in-distribution (ID) scenarios, while the posterior remains close to traffic-law compliant prior prediction on OOD scenarios. RuleFuser can be trained on nominal driving logs; it does not require exposure to any OOD scenarios at train time.

Statement of contributions: The main contributions of this work are as follows:

- We develop a novel method for injecting traffic rules in IL-based trajectory predictors by leveraging evidential deep learning approach.
- We introduce RuleFuser a transformer-based neural framework that uses dynamic input anchor splines in a joint encoder with a posterior-net style normalizing flow decoder to estimate both the epistemic and aleatoric uncertainty. We adopt the rule-based predictor in [2] and use its output as a prior in a novel fusion strategy to provide robustness against OOD scenarios.
- We demonstrated the ability of RuleFuser to adapt to OOD scenarios on a real-world nuPlan [3] AV dataset.

II. Related Work

Our approach combines the strengths of rule-based trajectory prediction and uncertainty-aware, learning-based trajectory prediction builds on a rich literature in both topics.

Traffic Rules in Trajectory Prediction. The behavior of road users is largely constrained by local traffic rules and customs. As such, it is appealing to leverage traffic law in trajectory prediction and planning. To do so requires both (i) *representing* traffic law, comprised of complex temporal rules as well as a variety exceptions depending on circumstances, as well as (ii) *utilizing* a representation of these rules in the trajectory prediction model.

Various representations of the traffic law have been explored in the literature, such as natural language [4, 5], formal logic formulae using metric temporal logic (MTL) [6, 7], linear temporal logic (LTL) [8, 9], and signal temporal logic [10, 11]. While effective in encoding rules for particular scenarios, these approaches struggle to generalize due to the countless

Jay Patrikar, and Sebastian Scherer are with Carnegie Mellon University {{jaypat,basti}@cmu.com}. Sushant Veer and Apoorva Sharma are are with NVIDIA Research {{sveer,apoorvas}@nvidia.com}. Marco Pavone is with Stanford University and NVIDIA Research {pavone@stanford.edu, mpavone@nvidia.com}.
*equal advising

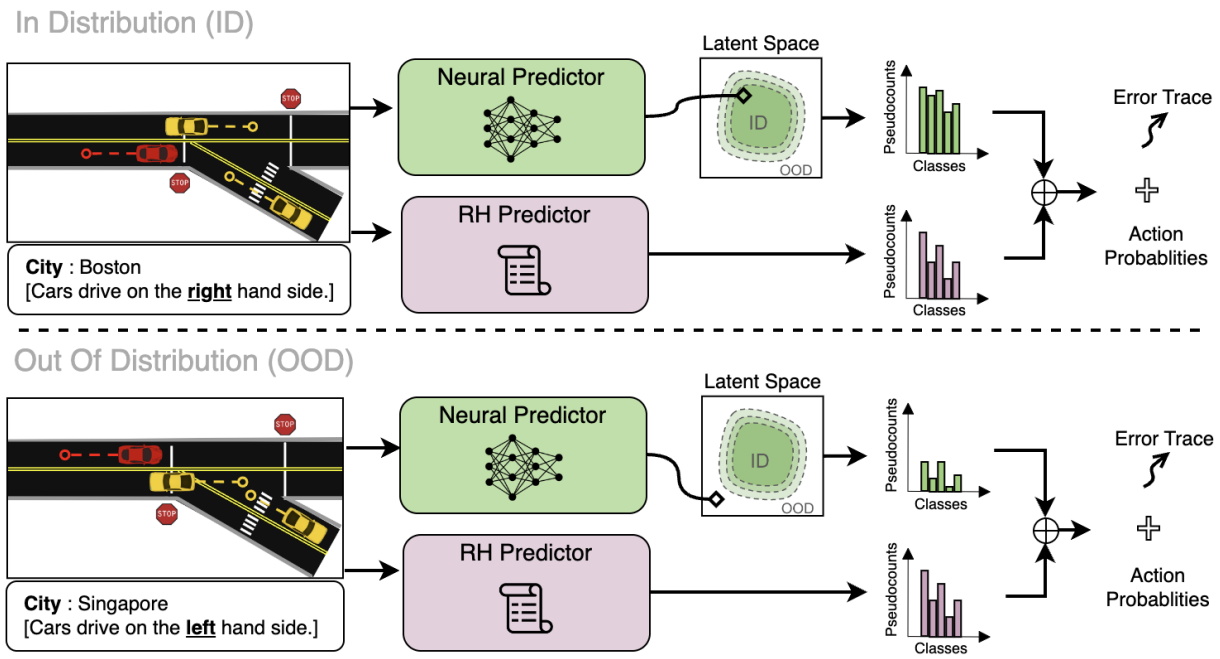

Fig. 1: Figure shows the overview of the RuleFuser approach. The framework uses a parallel setup with two predictors: a learned uncertainity-aware Neural Predictor and a rule-based RH Predictor. For in-distribution driving data, in this case Boston (top), the Neural Predictor learns to map the input to higher pseudo-counts areas in latent space. This provides a higher likelihood for the neural predictor leading to higher pseudo-counts. This leads to suppressing the contribution of the rule-based prior from the RH Predictor. For out-of-distribution scenarios, in this case Singapore (bottom), the input is mapped to lower pseudo-counts leading to larger influence of the rule-based prior.

exceptions that arise in real-world driving. To address this, prior work considers assigning an *importance order* to traffic rules, e.g. valuing avoiding collisions as more important than satisfying the speed limit [12, 13, 6, 14, 15].

In general, past work utilizing traffic law representation in trajectory prediction has been limited due to the generality / complexity trade-off of traffic rule representations. Early prediction models such as [16] can be interpreted as encoding the most basic of traffic rules; that of collision avoidance. More recently, researchers have considered using traffic rules to augment neural network prediction models, either as a loss function during training [17, 18, 19, 20], or as an output filter, using traffic rules to guide model sampling via Monte-Carlo tree search [21]. However, these works typically only leveraged a subset of traffic law. [2] demonstrated that scoring candidate trajectories using the scalar rank-preserving reward of [12] could serve as a capable trajectory predictor, and proposed an online filtering scheme which switched between a rule-based and learning-based model by monitoring their past prediction performance. In this work, we build on this work by leveraging strong uncertainty quantification to proactively interpolate between learned and rule-based predictions, without needing to make and observe poor predictions first.

Uncertainty Quantification in Trajectory Prediction. Predicting the future behavior of agents on the road is subject to both *aleatoric* and *epistemic* uncertainty. Aleatoric uncertainty captures the irreducible uncertainty present in the data: given the same scenario, agents may take different actions in the future, e.g., either continuing straight or turning right. For the most part, data-driven trajectory prediction methods have focused on quantifying aleatoric uncertainty by predicting a distribution over future behavior, rather than a single trajectory, and learning this distribution from data [22, 23]. Epistemic uncertainty, on the other hand, reflects uncertainty which in theory could be reduced, e.g. stemming from having limited training data for an unusual scenario. Bayesian methods, or approximations such as Monte-Carlo Dropout [24], Deep Ensembles [25], or variational inference based approaches [26, 27, 28, 29] aim to quantify epistemic uncertainty by modeling the distribution of models that are consistent with training data, but require additional computation beyond a single forward pass of the model. Other approaches aim to directly quantify epistemic uncertainty in the forward pass, by reasoning about deviations from training data, either in the input or latent space [30, 31, 32] or directly training the model to provide an estimate [33, 34, 35], or a combination [1]. Most related to our approach is [36], which applies the ideas in [1] to the problem of AV trajectory forecasting. A key limitation of all of these works is that, while they offer a signal to recognize when inputs to a neural prediction model may have high epistemic uncertainty and should not be trusted, their predictions are not grounded, and often depend on the random initialization of network weights. In this work, we address this limitation by leveraging a rules-based predictor as an informative prior which grounds predictions on out-of-distribution scenarios.

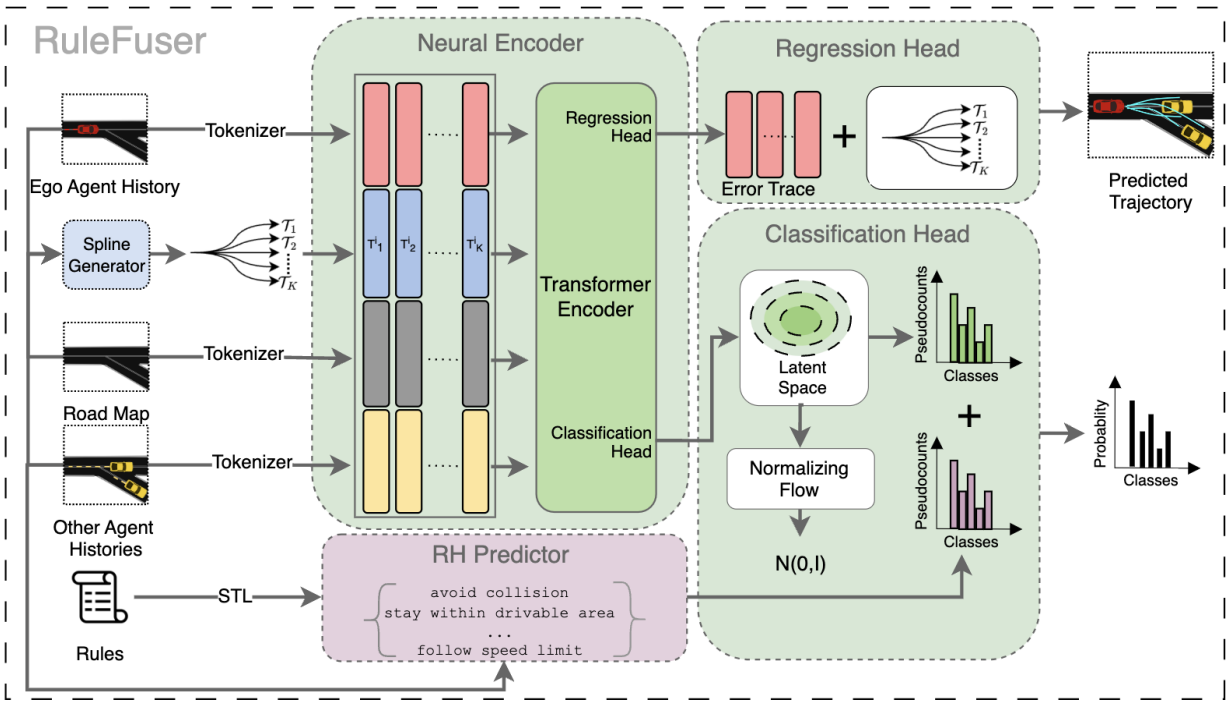

Fig. 2: Figure shows the implementation details for the RuleFuser framework. The Transformer-based encoder takes as input the history of the ego-agent, the scene and other traffic agents as well as a dynamically generated set of anchor trajectories. The encoder output is split into two heads. The regression head outputs the error trace for each of the anchor trajectories. The classification head uses a normalizing flow model to provide a confidence-aware Dirichlet distribution for anchor trajectories.

III. Problem Setup and Overview

Let $\boldsymbol{x} \in \mathcal{X}$ be the state of the ego agent, i.e., the agent that we are predicting for, $\boldsymbol{y} \in \mathcal{Y}$ be the joint state for all other traffic agents in the scene, and $\boldsymbol{s} \in \mathcal{S}$ be the map features. Our goal is ego trajectory prediction. Specifically, given the past behavior of the ego $\boldsymbol{x}_{t-H:t}$, other agents $\boldsymbol{y}_{t-H:t}$ and the map features $\boldsymbol{s}$, our goal is to predict the future ego behavior $\boldsymbol{x}_{t:t+F}$, where H is the length of the past context available, and F is the horizon length for prediction. In sum, we would like a model to predict $p(\boldsymbol{x}_{t:t+F} \mid \mathcal{H})$, where we use $\mathcal{H} := (\boldsymbol{x}_{t-H:t}, \boldsymbol{y}_{t-H:t}, \boldsymbol{s})$ as shorthand for all historical context available to the predictor.

In our approach, we structure the model's predictions by first generating a set of K anchor trajectories encoding possible futures $\{\mathcal{T}_k := \hat{\boldsymbol{x}}^k_{t:t+F}\}_{k=1}^K$ by exploiting the differential flatness of the bicycle dynamics model and fitting splines connecting the current ego state to different potential future states [37]. Choosing these splines reduces a high dimensional regression problem to a classification problem: now, our predictions take the form of a categorical distribution $\boldsymbol{q} \in \Delta^K$ over these anchor trajectories, where Δ^K represents the K-dimensional simplex. A rule-based planner is then used to furnish a prior $\mathbb{P}(\boldsymbol{q} \mid \mathcal{H})$ over Δ^K conditioned on the joint scene history $\boldsymbol{y}_{t-H:t}$ and map features $\boldsymbol{s}$.

How should we use our training data to update this prior? Suppose we treat each scene context independently, and that for a particular $\boldsymbol{y}_{t-H:t}$ and $\boldsymbol{s}$, we have N relevant examples in our training dataset. Let n_k represent the number of examples where anchor trajectory $\mathcal{T}_k$ corresponded to the true ego future, and let $\boldsymbol{n}$ be the vector of these counts. By Bayes rule,

$$\mathbb{P}(\boldsymbol{q} \mid \mathcal{H}, \boldsymbol{n}) \propto \mathbb{P}(\boldsymbol{q} \mid \mathcal{H}) \cdot \mathbb{P}(\boldsymbol{n} \mid \boldsymbol{q}, \mathcal{H}). \tag{1}$$

In this classification setting, $\mathbb{P}(\boldsymbol{n} \mid \boldsymbol{q}, \mathcal{H})$ is a multinomial distribution. The conjugate prior for a multinomial distribution is a Dirichlet distribution. This means that if we parameterize the prior as $\mathrm{Dir}(\boldsymbol{\beta}_{\text{prior}})$, then the posterior according to (1) will also be a Dirichlet distribution, $\mathrm{Dir}(\boldsymbol{\beta}_{\text{post}})$ where $\boldsymbol{\beta}_{\text{post}} = \boldsymbol{\beta}_{\text{prior}} + \boldsymbol{n}$. We can see that the number of matching examples $N = \|\boldsymbol{n}\|_1$, also known as the *evidence* controls the influence of the prior on the posterior prediction. Furthermore, this expression demonstrates why the parameter $\boldsymbol{\beta}_{\text{prior}}$ can be interpreted as *pseudocounts* for each class.

In reality, as $\boldsymbol{y}_{t-H:t}$ and $\boldsymbol{s}$ are continuous, we will not have any exact matching scenarios in our training dataset. However, scenarios are not independent from one another: training examples from similar scenes should influence our predictions on new scenes. Thus, while the Bayesian update logic above may not directly apply, we can nevertheless achieve a similar behavior by training a neural network model to estimate the evidence $\boldsymbol{n}$. In particular, we build on the approach of [1], and use a latent-space normalizing flow to estimate the evidence for each anchor trajectory. In this way, in-distribution $(\mathcal{H}, \mathcal{T}_k)$ pairs that are assigned high density in the normalizing flow receive higher evidence, while out-of-distribution pairs will be assigned lower evidence. As a result, if the scene is *more* OOD, then $\|\boldsymbol{n}\|$ is small and $\boldsymbol{\beta}_{\text{prior}}$ dominates the posterior

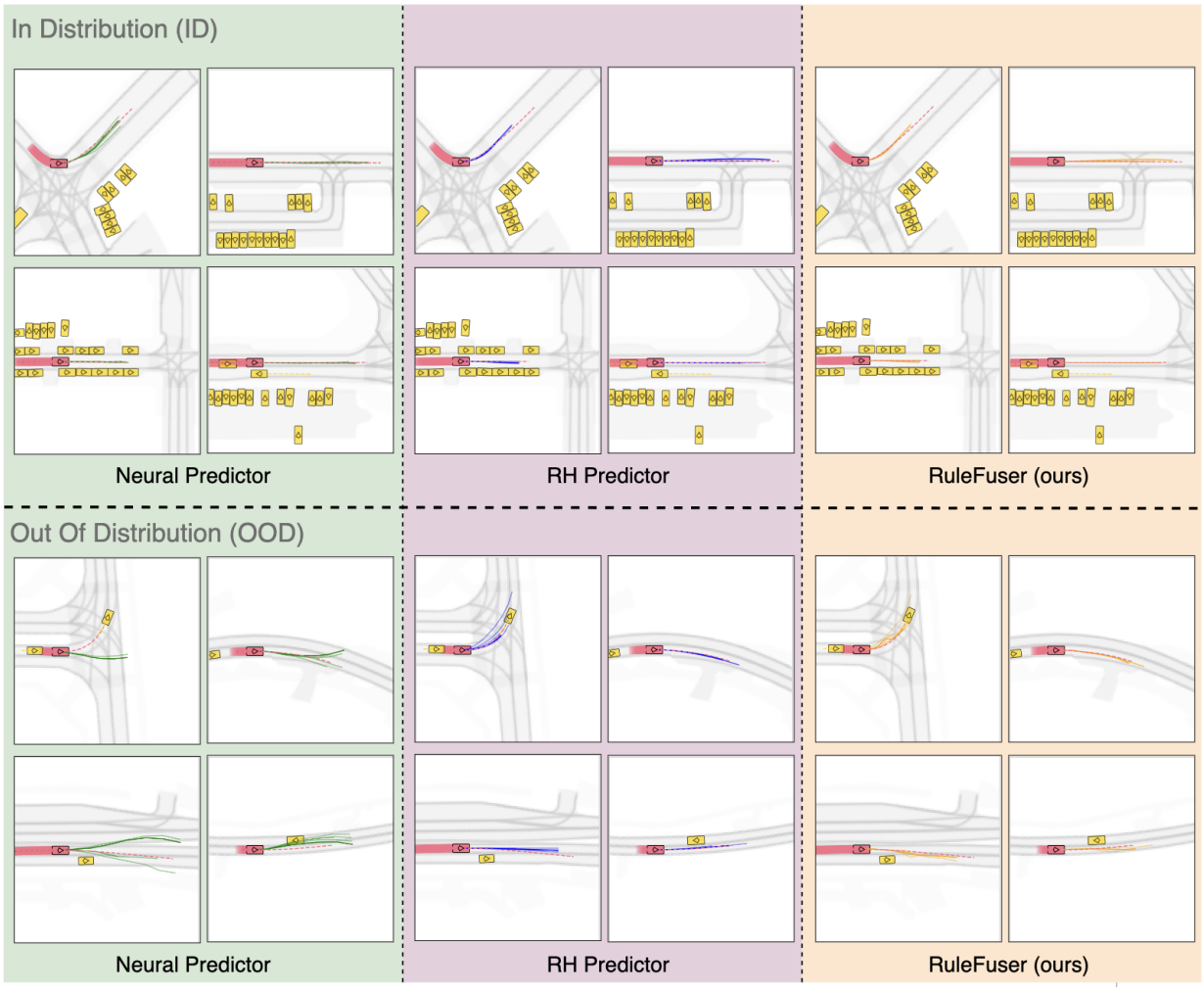

Fig. 3: Figure shows the qualitative results for the three methods. While the predicted trajectories using Neural Predictor showcase good performance in Boston (ID), the performance deteriorates in the Singapore (OOD). Rule-aware Predictor has consistent performance in both ID and OOD but fails to capture nuances in speed and turning radius. RuleFuser shows consistent performance in both ID and OOD scenarios preferring higher performance in ID and falling back to safety in OOD.

resulting in a smooth fallback to rules-based prediction, and if the scene is *more* ID, then $\boldsymbol{n}$ will have a larger magnitude, making the predictor rely more on the learnt network. Fig. 1 illustrates the framework for both ID and OOD scenarios.

IV. RuleFuser

In this section we introduce RuleFuser and describe its components; see Fig. 2 for an architectural overview of RuleFuser.

A. Rule-Hierarchy (RH) Predictor

The Rule-Hierarchy predictor, henceforth referred to as RH predictor, is tasked with predicting trajectories in compliance with a prescribed set of traffic rules. RH predictor is inspired from [2] and takes the form of a trajectory evaluator which takes in K anchor trajectories and provides a categorical distribution p over them. The categorical distribution p is then transformed into a Dirichlet prior $\mathrm{Dir}(p_{\text{prior}})$ by assigning psuedocounts in proportion to p.

Rules

We express the traffic rules in the form of a rule hierarchy [12] expressed in signal-temporal logic (STL) [38] using STLCG [39]. Rule hierarchies permit violation of the less important rules in favor of the more important ones if all rules cannot be simultaneously met. This flexibility allows for more human-like behaviors [40]. Our rule hierarchy consists of seven rules in decreasing order of importance:

- `avoid collision`
- `stay within drivable area`
- `follow traffic lights`
- `follow speed limit`
- `forward progression`
- `stay near the lane polyline`
- `stay aligned with the lane polyline`

Note that this 7-rule hierarchy expands on the one used in [2] which comprised of only four rules.

Route Predictor

The route predictor generates the reference polyline that we expect the agent to follow. Our route predictor plans to keep the agent in its current lane. If there are multiple branches upcoming in the lane, the route predictor chooses the branch that results in the smallest orientation deviation from the original lane.

Trajectory Evaluation

The rule hierarchy comes equipped with a scalar reward function R which measures how well a trajectory adheres to the specifications provided in the hierarchy. The exact specifics and construction of the reward function is beyond the scope of this paper – interested readers can find more details in [12].

Prior Computation

Let $\{R_1, \cdots, R_K\}$ be the reward for the K anchor trajectories. We transform these rewards into a Boltzmann distribution by treating the rewards as the negative of the Boltzmann energy. For each anchor trajectory i,

$$[\boldsymbol{p}]_k = \frac{\exp(R_k/\zeta)}{\sum_{k'=1}^{K} \exp(R_{k'}/\zeta)}, \tag{2}$$

where ζ is the Boltzmann temperature. Now, we use this distribution to define a prior by assigning pseudocounts in proportion to the probability assigned to each anchor trajectory. Let N_{prior} be a hyperparameter representing a budget on the counts. Then, we choose, $\boldsymbol{\beta}_{\text{prior}} := N_{\text{prior}} \cdot \boldsymbol{p}$ which is then used to define the prior as $\mathbb{P}(\boldsymbol{q} \mid \mathcal{H}) := \text{Dir}(\boldsymbol{\beta}_{\text{prior}})$.

B. Evidential Neural Predictor

This section provides details on the evidential transformer network. The network architecture is designed to take as input the position history of the ego agent $\boldsymbol{x}_{t-H:t}$, the position history of all other agents in the scene $\boldsymbol{y}_{t-H:t}$ as well as the scene geometry $\boldsymbol{s}$. In addition to this, the network also takes as input the set of K anchor trajectories $\{\mathcal{T}_k\}_{k=1}^{K}$. These anchor trajectories serve as a common reference between the neural and rule-based predictor. The network has two heads: A classification head leverages a normalizing flow to estimate the evidence n_k for each anchor trajectory $\mathcal{T}_k$. Furthermore, to allow the network to fine-tune the raw splines, a regression head produces an additive error trace $\boldsymbol{e}_{t:t+F}^{k}$ for each of the anchor trajectories. See Figure 2 for an overview. The posterior distribution over the modes is computed according to (1) using the estimated evidence $\boldsymbol{n}$.

Encoder

The encoder embeds each context-future pair $(\mathcal{H}, \mathcal{T}_k)$ into a latent space. First, we embed all inputs into a shared embedding space. The encoder first concatenates each of the anchor trajectory with the history of the ego agent. For each anchor trajectory $\mathcal{T}_k$, we construct the total ego trajectory candidate

$$\hat{\boldsymbol{x}}_{t-H:t+F}^{k} = [\boldsymbol{x}_{t-H:t}, \mathcal{T}_k] \tag{3}$$

This full trajectory is subsequently encoded using a single-layer MLP tokenizer to produce vector embeddings for each timestamp. The trajectory information of other agents in the scene is encoded via a separate MLP. Different number of traffic agents with different history lengths are padded to produce a single consistent embedding vector per historical timestep. The scene map information provided to the model as a rasterized map which is processed by a convolutional network followed by transformer layers.

Each $(\mathcal{H}, \mathcal{T}_k)$ pair is processed independently by the main transformer encoder. Having embedded all vectors into a shared embedding space, we concatenate them to create $\mathcal{C}_t^k$ which is the embedding per timestamp to produce a length $H + F$ sequence of embeddings for the transformer. For each input timestamp $t - H : t + F$, we concatenate the ego trajectory embedding, the corresponding social history embedding and the scene embedding to get $\mathcal{C}_t^k$. Future social embeddings are padded with zeros.

$$\mathcal{C}_t^k = \text{Embed}(\hat{\boldsymbol{x}}_t^k) \oplus \text{Embed}(\boldsymbol{y}_t) \oplus \text{Embed}(\boldsymbol{s}) \tag{4}$$

where $\oplus$ represents the concatenation operation. The output of the encoder is a processed sequence of embeddings $\mathcal{Z}$.

Decoder

The output of the encoder is decoded into two outputs.

(a) Regression Head: The regression head applies an MLP on the encoder outputs to compute an error trace $\boldsymbol{e}_{t:t+F}^{k}$ for each anchor trajectory $\mathcal{T}_k$. In this way, the learned network can fine-tune the mean of each of the modes of its prediction.

(b) Classification Head: The classification head aims to predict the evidence n_k to assign to each of the anchor trajectories. To do so, we leverage normalizing flows. Specifically, for each anchor trajectory, we first mean-pool the output of the encoder to obtain a single vector per anchor trajectory. Then, an MLP embeds this vector into a lower-dimensional space

$$\boldsymbol{z}_k = \text{MLP}(\text{MeanPool}(\mathcal{Z}_k)). \tag{5}$$

We model the evidence assigned to each mode as

$$n_k = N \cdot p_\psi(\boldsymbol{z}_k) \tag{6}$$

where N is the size of the training dataset and $p_\psi(\cdot)$ is estimated using a normalizing flow model with parameters ψ, following [1]. Note that the same normalizing flow model is applied to each anchor trajectory; this is because the anchor trajectories are dynamically generated as a function of ego state and map, and therefore the specific value of k has no semantic meaning. As $p_\psi(\cdot)$ is a valid probability distribution over the latent space, it cannot assign high likelihoods uniformly across the latent space. This encourages the model to produce low evidence for OOD scenarios, i.e. $(\mathcal{H}, \mathcal{T}_k)$ pairs that are not well represented in the training data.

The evidence for each mode is concatenated into the vector $\boldsymbol{n}$ over all anchor trajectories.

Fusion Strategy & Loss

Finally, our model fuses the outputs RH Predictor and the Evidential Neural Predictor to produce an ultimate probabilistic prediction for the future agent behavior. First, we apply Bayes rule using our estimated evidence $\boldsymbol{n}$ to compute the posterior $\mathbb{P}(\boldsymbol{q} \mid \mathcal{H}, \boldsymbol{n}) = \mathrm{Dir}(\boldsymbol{\beta}_{\mathrm{post}})$:

$$\boldsymbol{\beta}_{\mathrm{post}} = \boldsymbol{\beta}_{\mathrm{prior}} + \boldsymbol{n}. \tag{7}$$

Then, the posterior predictive distribution over the ego future behavior takes the form of a Dirichlet-Normal distribution:

$$\boldsymbol{q} \sim \mathrm{Dir}(\boldsymbol{\beta}_{\mathrm{post}}) \tag{8}$$

$$k \mid \boldsymbol{q} \sim \mathrm{Cat}(\boldsymbol{q}) \tag{9}$$

$$\boldsymbol{x}_{t:t+F} \mid k \sim \mathcal{N}(\mathcal{T}_k + \boldsymbol{e}^k_{t:t+F}, \boldsymbol{I}), \tag{10}$$

where Cat represents the categorical distribution and $\mathcal{N}$ represents the normal distribution. Marginalizing over the realization of $\boldsymbol{q}$, the final predictive distribution of our model can be viewed as a Mixture of Gaussians:

$$\bar{\boldsymbol{q}} = \mathbb{E}[\boldsymbol{q}] = \boldsymbol{\beta}_{\mathrm{post}} / (\mathbf{1}^T \boldsymbol{\beta}_{\mathrm{post}}) \tag{11}$$

$$p(\boldsymbol{x}_{t:t+F} \mid \mathcal{H}) = \sum_{k=1}^{K} \bar{p}_k \cdot \mathcal{N}(\boldsymbol{x}_{t:t+F}; \mathcal{T}_k + \boldsymbol{e}^k_{t:t+F}, \boldsymbol{I}) \tag{12}$$

To train this model, we follow prior work on training mixture models for trajectory prediction and train for classification and regression independently.

Specifically, we impose a regression loss using a masked mean-square-error (MSE) loss function:

$$k^* = \arg\min_k ||\mathcal{T}_k - \boldsymbol{x}_{t:t+F}||_2^2, \tag{13}$$

$$\mathcal{L}_{MSE} = \sum_{k=1}^{K} \mathbf{1}_{k=k^*} \cdot ||(\boldsymbol{e}^k_{t:t+F} + \mathcal{T}_k) - \boldsymbol{x}_{t:t+F}||_2^2. \tag{14}$$

To train the classification head, we follow [1] and augment a classification loss on the marginal prediction with a penalty on assigning evidence to incorrect modes. Specifically, we use a binary cross entropy loss (independently over the anchor trajectories), with an entropy reward to discourage assigning evidence to incorrect classes:

$$\mathcal{L}_{UCE} = \sum_{k=1}^{K} \mathrm{BCE}(\bar{q}_k, \mathbf{1}_{k=k^*}) - H(\boldsymbol{\beta}_{\mathrm{post}}) \tag{15}$$

where BCE is the binary cross entropy loss, and $H(\boldsymbol{\beta}_{\mathrm{post}})$ is the entropy of the Dirichlet distribution.

The parameters of the embedding networks, transformer encoder, decoder networks, and normalizing flow layers are all optimized through stochastic gradient descent to optimize a weighted sum of $\mathcal{L}_{MSE}$ and $\mathcal{L}_{UCE}$.

V. Experiments

In this section, we demonstrate the ability of RuleFuser to deliver good prediction performance while better adhering to traffic rules for both ID and OOD scenes.

A. Datasets

In order to test the efficacy of the model we use the NuPlan trajectory prediction dataset. NuPlan provides labeled data for multiple cities around the world. Labeling individual data points that are OOD is prohibitively expensive and often subjective in a large dataset. Instead, for this work, we choose to focus on two cities: Boston and Singapore. The choice is motivated by the fact that due to local laws, drivers in Boston drive on the right-side of the road while drivers in Singapore drive on the left-side. This provides an ideal setup wherein the model is trained on the train split of the Boston dataset and tested on the test splits of both Boston (ID) and Singapore (OOD) to study RuleFuser's performance in both ID and OOD.

B. Predictors

We compare the performance of three models: Neural predictor, RH predictor, and RuleFuser. Neural predictor mirrors the architecture of RuleFuser differing only in the choice of the prior on the anchor trajectories which is always set to be uniform. Effectively, the neural predictor is "blind" to the traffic rules and relies solely on the training data. The RH predictor is the rules-based predictor described in Section IV-A that operates exclusively according to user-defined traffic rules without utilizing data. RuleFuser integrates both the above, as described in Section IV, leveraging both the training data and user-defined traffic rules.

C. Metrics

We use multiple metrics to compare the performance of the Neural and RH predictors and RuleFuser. We separate metrics into two categories: performance and safety.

Performance Metrics

- mADE_k: Minimum Average Displacement Error is the smallest average Euclidean distance between the top_k predicted trajectories (according to the predictor) and the ground truth trajectory.
- mFDE_k: Minimum Final Displacement Error is the smallest Euclidean distance between the terminal positions of the top_k predicted trajectories (according to the predictor) and the terminal position of the ground truth trajectory.
- pADE: Probability Weighted Average Displacement Error is the probability-weighted sum of the Euclidean distance between all K predicted trajectories and the ground truth trajectory. The probabilities here refer to the categorical distribution on the anchor trajectories outputted by the predictors.
- pFDE: Probability Weighted Final Displacement Error is the probability-weighted sum of the Euclidean distance between the terminal positions of all K predicted trajectories and the ground truth trajectory. The probabilities here refer to the categorical distribution on the anchor trajectories outputted by the predictors.
- Acc.: Accuracy is the fraction of predicted class labels that match the ground truth class label.
- KLDiv.: Kullback-Leibler divergence, or relative entropy, is the KL divergence between the predicted categorical

Metrics	mADE/mFDE ↓			pADE/pFDE ↓	Acc. ↑	KLDiv. ↓	NLL↓	Evidence ($\mathbf{1}^T\boldsymbol{n}$)
top$_k$	1	5	20	All	All	All	All	All
Boston (ID):								
Neural Predictor	0.51/1.31	0.34/0.77	0.29/0.57	0.55/1.42	0.53	4.56	1.19	–
RH Predictor	1.55/4.00	0.91/2.25	0.57/1.35	1.66/4.32	0.08	2.60	1.29	–
RuleFuser	0.56/1.44	0.35/0.80	0.29/0.57	0.71/1.87	0.52	4.14	1.19	7.8e+10
Singapore (OOD):								
Neural Predictor	0.92/2.41	0.57/1.40	0.40/0.85	0.96/2.53	0.28	3.90	1.52	–
RH Predictor	1.41/3.61	0.79/1.93	0.54/1.24	1.68/4.36	0.12	2.73	1.58	–
RuleFuser	0.97/2.56	0.55/1.33	0.39/0.80	1.1/3.04	0.29	3.43	1.52	5.4e+10

TABLE I: Table shows the quantitative performance metrics for the Neural Predictor, RH Predictor, and RuleFuser on nuPlan's Boston (ID) and Singapore (OOD) datasets. The predicted trajectories have a horizon of 4 seconds.

Metrics	% Collision Rate ↓			% OffRoad Rate ↓		
top$_k$	1	5	20	1	5	20
Boston (ID):						
Neural Predictor	4.2	5.4	7.4	5.6	6.6	8.0
RH Predictor	2.4	2.5	3.3	2.3	2.7	3.9
RuleFuser	3.9	4.5	6.9	5.0	5.0	7.0
Singapore (OOD):						
Neural Predictor	3.1	3.8	5.0	6.6	8.2	11.7
RH Predictor	1.4	1.4	1.7	0.7	0.8	2.2
RuleFuser	2.5	2.7	4.4	3.0	3.3	9.2

TABLE II: Table shows the results on quantitative safety metrics for the three predictors on both the Boston (ID) and Singapore (OOD).

distribution on the anchor trajectories and the one-hot ground truth categorical distribution.

- NLL.: Negative Log Likelihood under the marginal predictive distribution (12)

Safety Metrics

- Percentage Collision Rate: This metric measures the percentage of times the predicted trajectory indicated a collision with the ground truth future trajectory of another traffic agent.
- Percentage Off-Road Rate: This metric measures the percentage of times the predicted trajectory led to an off-road event.

D. Discussion

Our experiments provide two key findings: (i) RuleFuser achieves better prediction performance than the RH predictor and better traffic rule satisfaction than the Neural Predictor, striking a better balance between prediction performance and traffic rule compliance; (ii) RuleFuser provides greater gains in OOD scenarios. In the rest of this section we will elaborate more on these findings.

RuleFuser balances performance and rule satisfaction.

Inspecting Table I shows that the Neural Predictor and RuleFuser perform very well on prediction metrics while the RH Predictor's performance is significantly worse, which can be attributed to the RH Predictor's inability to correctly infer the intent of the agent we are predicting for [2, Fig. 3]. On the other hand, in Table II, the RH Predictor generates the most rule adherent trajectories, followed by RuleFuser and then the Neural Predictor. The key point to note here is that the prediction performance of RuleFuser is very similar (albeit slightly degraded) to that of the Neural Predictor while its performance on the safety metrics is significantly better than the Neural Predictor.

Impact of RuleFuser in OOD.

As expected from a learning-based predictor, the neural predictor performs well on ID scenarios in the Boston dataset, while its performance degrades in OOD scenarios. This leads to greater volume of off-road and collision trajectories. On the other hand, RH predictor showcases consistent performance on both prediction and safety metrics in both ID and OOD scenarios. RuleFuser demonstrates a balance between neural and RH predictors by preferring the higher performance neural predictor in ID scenarios but falling back to higher safety RH predictor in OOD scenarios. Consequently, RuleFuser's performance on the safety metrics shows a greater improvement in the OOD scenarios than in ID scenarios. The improvement is more significant in the offroad rate than for collision rate because the OOD dataset (Singapore) has much lower traffic density than the ID dataset (Boston); Singapore dataset has an average of 3.23 neighboring vehicles per scene while the Boston dataset has an average of 11.26 neighboring vehicles per scene. Finally, we note that between ID and OOD datasets, the total evidence $\mathbf{1}^T\boldsymbol{n}$ shows lower counts in OOD than ID validating the ability of our normalizing-flow-based OOD detection scheme.

With all of these results, it is important to keep in mind that while the Singapore data represents data that is drawn from a different distribution, not every scenario will be significantly different from those seen in the Boston dataset. For example, the behavior a vehicle going straight and following a lane is easy to predict and behaves similarly across Boston and Singapore. Thus, not every scenario in the Singapore dataset will be outside the domain of competency of the Neural Predictor. For this reason, even on the Singapore split, we do not expect the evidence to be zero – indeed, our results show that RuleFuser does not assign zero evidence, and generalizes

better to the OOD dataset (in terms of top-5 and top-20 mADE and mFDE) than either of the Neural Predictor and the RH Predictor alone.

VI. Conclusion and Future Work

This work presents RuleFuser, a novel trajectory prediction framework that strikes a balance between performance and traffic rule compliance. RuleFuser accomplishes this by modeling trajectory prediction as an evidential regression. The RH predictor supplies a prior on the future trajectories to a neural evidential predictor. The neural predictor generates its own predictions and an OOD measure for scene using normalizing flows to produce a posterior distribution on the future trajectories; the more OOD the scene, the more the posterior tends towards the prior. Using real-world AV datasets, we demonstrate the ability of RuleFuser to leverage data-driven predictions when it encounters an ID scene while relying more on "safer" rule-based predictions in OOD scenes.

This work opens up various exciting future directions: First, we will leverage RuleFuser to develop a traffic rule aware learning-based motion planner. Second, we note that RuleFuser delivers near state-of-the-art (sota) prediction performance [41] despite being trained only on the Boston split of the nuPlan dataset. RuleFuser holds the potential to establish a new sota for trajectory prediction. Finally, we will expand on our experiments to more deeply understand the performance of RuleFuser in OOD scenes.

Acknowledgments

This work used Bridges-2 at PSC through allocation cis220039p from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #213296. The authors would also like to thank Ingrid Navarro for her help in formulating the social and map tokenizers.

References

[1] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without OOD samples via density-based pseudo-counts. *Advances in Neural Information Processing Systems*, 33: 1356–1367, 2020.

[2] Sushant Veer, Apoorva Sharma, and Marco Pavone. Multi-predictor fusion: Combining learning-based and rule-based trajectory predictors. *arXiv preprint arXiv:2307.01408*, 2023.

[3] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition ADP3 workshop*, 2021.

[4] Boyi Li, Yue Wang, Jiageng Mao, Boris Ivanovic, Sushant Veer, Karen Leung, and Marco Pavone. Driving everywhere with large language model policy adaptation. *arXiv preprint arXiv:2402.05932*, 2024.

[5] Kumar Manas, Stefan Zwicklbauer, and Adrian Paschke. Robust traffic rules and knowledge representation for conflict resolution in autonomous driving. In *Proc. of the 16th International Rule Challenge and 6th Doctoral Consortium @ RuleML+RR*, 2022.

[6] Kumar Manas and Adrian Paschke. Legal compliance checking of autonomous driving with formalized traffic rule exceptions. In *Proc. of the Workshop on Logic Programming and Legal Reasoning in conjunction with 39th International Conference on Logic Programming (ICLP)*, 2022.

[7] Sebastian Maierhofer, Anna-Katharina Rettinger, Eva Charlotte Mayer, and Matthias Althoff. Formalization of interstate traffic rules in temporal logic. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 752–759. IEEE, 2020.

[8] Klemens Esterle, Luis Gressenbuch, and Alois Knoll. Formalizing traffic rules for machine interpretability. In *Proc. of IEEE Connected and Automated Vehicles Symposium (CAVS)*, pages 1–7. IEEE, 2020.

[9] Jesper Karlsson and Jana Tumova. Intention-aware motion planning with road rules. In *Proc. of IEEE Int. Conf. on Automation Science and Engineering*, pages 526–532. IEEE, 2020.

[10] Federico Pigozzi, Eric Medvet, and Laura Nenzi. Mining road traffic rules with signal temporal logic and grammar-based genetic programming. *Applied Sciences*, 11(22): 10573, 2021.

[11] Mohammad Hekmatnejad, Shakiba Yaghoubi, Adel Dokhanchi, Heni Ben Amor, Aviral Shrivastava, Lina Karam, and Georgios Fainekos. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *Proc. of the ACM-IEEE Int. Conf. on Formal Methods and Models for System Design*, pages 1–11, 2019.

[12] Sushant Veer, Karen Leung, Ryan Cosner, Yuxiao Chen, Peter Karkus, and Marco Pavone. Receding horizon planning with rule hierarchies for autonomous vehicles. *arXiv preprint arXiv:2212.03323*, 2022.

[13] Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *Proc. IEEE Conf. on Robotics and Automation*, pages 8536–8542, 2019.

[14] Cristian-Ioan Vasile, Jana Tumova, Sertac Karaman, Calin Belta, and Daniela Rus. Minimum-violation scltl motion planning for mobility-on-demand. In *Proc. IEEE Conf. on Robotics and Automation*, pages 1481–1488. IEEE, 2017.

[15] Jana Tůmová, Luis I Reyes Castro, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Minimum-violation ltl planning with conflicting specifications. In *Proc. American Control Conference*, pages 200–205, 2013.

[16] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[17] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. Dsdnet: Deep structured self-driving network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 156–172. Springer, 2020.

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

[19] Chris Paxton, Vasumathi Raman, Gregory D Hager, and Marin Kobilarov. Combining neural networks and tree search for task and motion planning in challenging environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 6059–6066. IEEE, 2017.

[20] Jiaxin Liu, Wenhui Zhou, Hong Wang, Zhong Cao, Wenhao Yu, Chengxiang Zhao, Ding Zhao, Diange Yang, and Jun Li. Road traffic law adaptive decision-making for self-driving vehicles. In *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, pages 2034–2041. IEEE, 2022.

[21] Jasmine Jerry Aloor, Jay Patrikar, Parv Kapoor, Jean Oh, and Sebastian Scherer. Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains. In *Proc. IEEE Conf. on Robotics and Automation*, pages 1320–1326. IEEE, 2023.

[22] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Proc. European Conf. on Computer Vision*, pages 683–700. Springer, 2020.

[23] Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Scept: Scene-consistent, policy-based trajectory predictions for planning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 17103–17112, 2022.

[24] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015.

[25] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Conf. on Neural Information Processing Systems*, 2017.

[26] Apoorva Sharma, Navid Azizan, and Marco Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. *arXiv preprint arXiv:2102.12567*, 2021.

[27] Hippolyt Ritter, Aleksandar Botev, and D. Barber. A scalable laplace approximation for neural networks. *Int. Conf. on Learning Representations*, 2018.

[28] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. *Int. Conf. on Machine Learning*, 2015.

[29] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3), 1992.

[30] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Conf. on Neural Information Processing Systems*, 2018.

[31] Ev Zisselman and Aviv Tamar. Deep residual flow for out of distribution detection. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[32] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *Int. Conf. on Learning Representations*, 2019.

[33] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9690–9700. PMLR, 13–18 Jul 2020.

[34] Andrey Malinin, Sergey Chervontsev, Ivan Provilkov, and Mark Gales. Regression prior networks. *arXiv preprint arXiv:2006.11590*, 2020.

[35] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Conf. on Neural Information Processing Systems*, 31, 2018.

[36] Masha Itkina and Mykel Kochenderfer. Interpretable self-aware neural networks for robust trajectory prediction. In *Proc. Conf. on Robot Learning*, pages 606–617. PMLR, 2023.

[37] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Proc. IEEE Conf. on Robotics and Automation*, pages 3399–3406, 2018.

[38] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer Berlin Heidelberg, 2004.

[39] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *arXiv preprint arXiv:2008.00097*, 2020.

[40] Bassam Helou, Aditya Dusi, Anne Collin, Noushin Mehdipour, Zhiliang Chen, Cristhian Lizarazo, Calin Belta, Tichakorn Wongpiromsarn, Radboud Duintjer Tebbens, and Oscar Beijbom. The reasonable crowd: Towards evidence-based and interpretable models of driving behavior. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 6708–6715, 2021.

[41] Yuxiao Chen, Sander Tonkens, and Marco Pavone. Categorical traffic transformer: Interpretable and diverse behavior prediction with tokenized latent. *arXiv preprint arXiv:2311.18307*, 2023.