Safe Real-World Autonomous Driving by Learning to Predict and Plan with a Mixture of Experts

Stefano Pini[1], **Christian S. Perone**[1], **Aayush Ahuja**[2], **Ana Sofia Rufino Ferreira**[2], **Moritz Niendorf**[2], **Sergey Zagoruyko**[1]

[1]Woven Planet United Kingdom Limited
[2]Woven Planet North America, Inc.
{firstname}.{lastname}@woven-planet.global

Abstract

The goal of autonomous vehicles is to navigate public roads safely and comfortably. To enforce safety, traditional planning approaches rely on handcrafted rules to generate trajectories. Machine learning-based systems, on the other hand, scale with data and are able to learn more complex behaviors. However, they often ignore that agents and self-driving vehicle trajectory distributions can be leveraged to improve safety. In this paper, we propose modeling a distribution over multiple future trajectories for both the self-driving vehicle and other road agents, using a unified neural network architecture for prediction and planning. During inference, we select the planning trajectory that minimizes a cost taking into account safety and the predicted probabilities. Our approach does not depend on any rule-based planners for trajectory generation or optimization, improves with more training data and is simple to implement. We extensively evaluate our method through a realistic simulator and show that the predicted trajectory distribution corresponds to different driving profiles. We also successfully deploy it on a self-driving vehicle on urban public roads, confirming that it drives safely without compromising comfort. The code for training and testing our model on a public prediction dataset and the video of the road test are available at `https://woven.mobi/safepathnet`.

1 Introduction

In the last decade, Autonomous Driving (AD) has been largely explored by researchers in academia and industry. Against expectations of many in the field, Self-Driving Vehicles (SDVs) are still constrained to limited operational domains and not yet ready to be deployed at scale. Among the AD stack, the weak link appears to be the planning module, which is where most decision-making takes place. This component not only has to reason about other road actors' actions and cover different driving behaviors, it must also guarantee safety and robustness against the long tail distribution of driving scenarios.

Traditional rule-based systems [19] addressed the planning task by defining progressively larger sets of hand-crafted rules, but they proved to scale poorly to complex or unfamiliar driving scenarios. While several data-driven approaches [34, 3, 35, 26, 9] were presented in recent years to improve scalability and performance by leveraging large datasets, they either approach planning as a unimodal trajectory forecasting problem, lack safety checks, or are computationally inefficient. Recently, [32] presented a hybrid two-stage approach consisting of a ML-based planner for trajectory forecasting and a safety fallback layer. However, the method relies on both an external prediction module and a rule-based trajectory generator that does not learn nor improve with data.

Machine Learning for Autonomous Driving Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, USA.

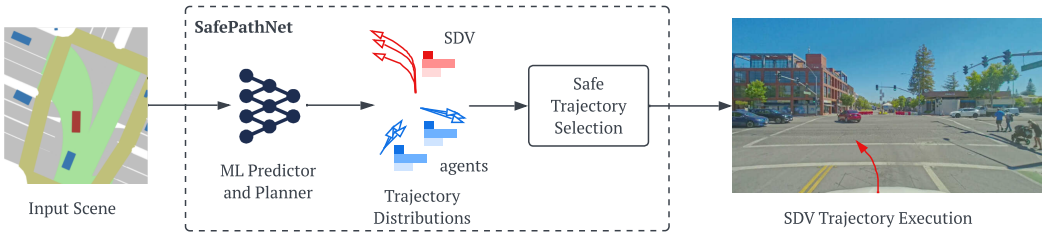

Figure 1: High-level overview of SafePathNet, a ML approach improving on-road safety of self-driving vehicles (SDVs). A neural network receives a vectorized scene representation, combining map, road agent and SDV states. It then predicts a set of future SDV driving plans (in red) and a set of agent future trajectories (in blue) with associated probabilities. A safe driving trajectory is selected given this data, and executed by the SDV in the real world.

In this paper, we propose SafePathNet, a ML-based prediction and planning system that leverages data uncertainty and scales with training data, while ensuring a safety profile similar to previous methods. SafePathNet jointly models prediction and planning as distributions of future trajectories, and leverages predictions to improve safety at inference time. Exploiting the powerful attention mechanism of the Transfomer module [31], we propose a deep neural network architecture that predicts diverse SDV's trajectories and other agents' future locations given a vectorized representation of the scene. To model data uncertainty, we use a Mixture of Experts (MoE) approach [15] that learns a distribution over trajectories. To improve safety of the SDV's plan, we perform cost-sensitive selection from the trajectory distribution according to the road agents' predicted locations to avoid collisions. This approach to safety is efficiently performed within the model itself, scales with data, and does not rely on any additional external signals, as SafePathNet selects the trajectory from the predicted distribution. Simulation and on-road tests show that the proposed approach models and exploits different driving profiles to improve planning safety without substantially impacting comfort.

In summary, our contributions are:

- We propose to model the distribution of future trajectories of agents and the SDV using a mixture of experts in a unified neural network for prediction and planning;
- We present an efficient and easy to implement decision-making approach that leverages the predicted trajectories and associated probabilities to improve safety by reducing collisions between the SDV and other road agents;
- We extensively validate our proposal in a realistic closed-loop simulator and deploy it on an SDV driving on public roads, confirming its effectiveness and safety;
- The code and hyperparameters for training and testing our model on a public prediction dataset will be made publicly available.

To the best of our knowledge, this is the first paper that combines a MoE approach with a decision-making strategy that uses the predicted distribution to drive safely on public roads (Figure 1).

2 Related work

Although many data-driven systems have been recently developed [3, 11] exploiting breakthroughs in Deep Learning methods, many challenges remain open for the deployment of these systems in the real world. Deep learned systems can scale and improve with data and are increasingly favored over hand-engineered approaches that do not scale in engineering effort. However, they also present new challenges, e.g. how to embed causal relationships and behaviors such as collision avoidance.

Data-driven approaches to planning can be grouped according to their training paradigm, such as Imitation Learning (IL) [36] and Reinforcement Learning (RL) [29]. Among IL methods, Behavioral Cloning [2] is one of the most used approaches of imitation, dating back to ALVINN [22] and showing recent developments such as in [3, 35, 32, 12, 5]. Although imitation approaches showed significant progress, the covariate shift induced by the policy is still an open problem that can make the model perform poorly during deployment. On the other hand, Reinforcement Learning [29] has reached many milestones on a large set of simulated environments. However, its uptake in

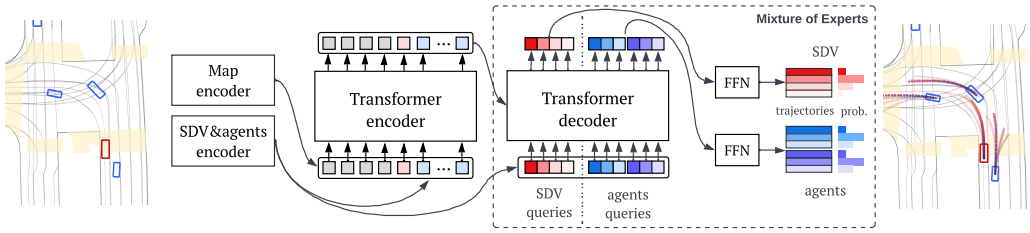

Figure 2: SafePathNet predicts SDV and road agents future trajectories given a vectorial representation of input scene. Firstly, map, SDV and agent features are encoded independently by PointNet-like networks. Then, their outputs are processed by a Transformer encoder-decoder network. Finally, decoder outputs are processed by FFNs predicting SDV and agent future trajectories together with probability distributions. Additionally, we use a kinematic model after SDV FFN, omitted for clarity.

real-world problems has been slower than expected [20]. While several methods have been proposed in recent years [28, 24, 17], they only make use of simulated environments or show very limited and constrained private-road testing, limiting their application in the real world. PredictionNet [16] show on-road tests too, but the RL policy only controls the SDV longitudinal speed. When tested in the real world, most of the aforementioned approaches make mistakes, mainly due to covariate shift. For this reason, safety guarantees are paramount for deployment in real scenarios. The most similar approach to our method is SafetyNet [32], where a two-stage pipeline is used to perform rule-based safety checks over a ML-based trajectory. SafetyNet depends on an external rule-based planner based on [33] and does not scale with more data as the trajectory generator is not learned. On the contrary, our approach learns to predict diverse trajectories from data, and exploits the SDV and road agent predictions to improve the safety of the planner.

Other works [10, 21, 8, 9, 30, 16, 1] propose the generation of multiple trajectories in the context of the SDV motion planning, localization, or road agent prediction. Some of these works [10, 21] produce predictions independently of each other while others [8, 9] produce scene-level consistent predictions. While the former approaches can generate unsafe SDV trajectories colliding with other road agents, the latter often suffer from sample inefficiency due to the large set of possible futures. In comparison, our approach simultaneously predicts a diverse set of SDV and road agent trajectories and associated probabilities and leverages them to increase the safety of the planner during inference in real-time.

3 Methodology

In this section, we first present the multimodal architecture and the training procedure we use to jointly predict future trajectories of other road agents and plans of the SDV. Then, we present a simple-yet-effective approach to improve safety of the planner during inference. We propose a cost-sensitive selection over the predicted SDV trajectory distribution exploiting road agents future locations to avoid collisions with them.

3.1 Model architecture

We address the joint tasks of prediction and planning with the neural architecture shown in Figure 2, inspired by previous approaches [11, 32]. Differently from such prior work, which mostly use unimodal predictions, we model the multimodal trajectory distribution using a Mixture of Experts (MoE) approach. That is, the deep model predicts different trajectory candidates and a probability distribution over them. We can then take advantage of these planning alternatives defining a selection policy to improve driving safety at inference time.

Input and output representation. We represent the driving scene in a vectorized format, and use the following data as input: (i) SDV as current pose, speed, acceleration, size, moving/stationary history for K_s seconds; (ii) road agents as pose, size, vehicle type for the current frame and the previous K_a seconds; (iii) static HD map including lanes, crosswalks, intersections; (iv) dynamic map elements including other non-moving obstacles and status of traffic lights; (v) goal (route) as the center of the

lane that the SDV should follow. Each input element points are encoded in SDV-centric reference frame and include the element type as additional feature.

The model outputs can be split in planning and prediction output. The planning output is composed of N SDV future trajectories τ^i and one probability distribution $p_i = p(\tau^i \mid \mathbf{x})$ over them. Each SDV trajectory τ^i is defined as a set of T_s discrete states

$$\tau_t^i = \{x_t^i, y_t^i, \theta_t^i, v_t^i, a_t^i, k_t^i, j_t^i\}, \tag{1}$$

where t represents a timestep in the range $[1, T_s]$, x, y, θ the pose, v the speed, a the acceleration, k the curvature, and j the jerk. In practice, the model outputs the jerk and curvature k_t^i, j_t^i for each timestep t and the remaining trajectory features are inferred using a unicycle kinematic vehicle model from the initial state x_0^i, y_0^i, θ_0^i. The probability distribution $p_i = p(\tau^i \mid \mathbf{x})$ is defined over the N SDV trajectories τ^i and can be use to pick the most appropriate one given the current input. We discuss this in detail in Section 3.2.

The prediction output is composed of $A \times M$ road agents' future trajectories ν_a^j, $j = 1, \dots M$ and A probability distributions $q_j^a = p(\nu_a^j \mid \mathbf{x})$ over each set of agent M future trajectories ν_a^j. Each road agent trajectory ν^j is instead defined as a set of T_a discrete states

$$\nu_t^j = \{x_t^j, y_t^j, \theta_t^j\}, \tag{2}$$

where t represents a timestep in the range $[1, T_a]$ and x, y, θ represent the pose. Similarly to the planning output, each probability distribution q^a, $a = 1, \dots A$ is defined over the M trajectories of the a-th agent. The distribution can be used to pick the most appropriate trajectory for each agent.

Architectural details. The architecture of SafePathNet is similar to those of VectorNet [11] and DETR [7], combining an element-wise point encoder [23] and a Transformer [31] (Fig. 2). The element-wise point encoder consists of two PointNet-like modules that are used to compress each input element from a set of points to a single feature vector of the same size. A series of Transformer Encoder layers are used to model the relationships between all input elements (SDV, road agents, static and dynamic map, route), encoded by the point encoder. Then, a series of Transformer Decoders are used to query SDV and agents features. We make use of a set of learnable embeddings to construct the queries of the Transformer Decoders. The SDV embeddings from the point encoder are added to the set of N learnable embeddings to obtain a different query for each SDV future trajectory that we aim to predict. Similarly, M learnable query embeddings are used to obtain a variable number of M different queries for each road agent. This architecture closely resembles DETR object detection network, fitting our set prediction task well. Here, each query embedding can encode a specific driving behavior corresponding to one expert of the MoE approach.

Finally, an SDV-specific decoder (FFN) converts each SDV feature to a set of control inputs (i.e. jerk, curvature) and a kinematic decoder translates them into a future trajectory. Similarly, an agent-specific decoder converts each agent feature to a future trajectory. In addition to trajectories, the decoder predicts a logit for each SDV and agent trajectory. For each element, the corresponding logits are converted to a probability distribution over the future trajectories by applying a softmax function. All road agents and SDV are modeled independently, but predicted jointly in parallel.

Training procedure. We adopt imitation learning and define our training objective as minimizing a distance between predicted SDV poses and the ground truth expert trajectories. Similarly, we minimize distance between predicted and ground truth agents' future trajectories. We additionally regularize jerk and curvature corresponding to SDV plans.

Our model represents a MoE and predicts multiple trajectories for the SDV and each road agent, corresponding to N/M experts, and a probability distribution over each trajectory set, corresponding to expert selection. To train the experts and expert selection jointly while avoiding mode collapse, we use a winner-takes-all approach, somewhat similar to previous methods [10]. Similarly to DETR [7], we formulate a matching cost between predicted and target trajectories and probabilities, making the expert with minimal cost the winner. Without loss of generality, in the following we describe the loss applied to one sample of the SDV planning, which is similarly applied to agent prediction too.

In details, we compute matching cost for each trajectory then select one trajectory according to

$$i^* = \arg\min_i \ \mathcal{L}_{\text{IL}}^i + \lambda(1 - p_i), \quad \text{where} \ \ \mathcal{L}_{\text{IL}}^i = \sum_{t=1}^{T} ||\tau_t^i - \hat{\tau}_t||_1 + \beta \mathcal{L}_{\text{reg}}^i, \tag{3}$$

where p_i is the predicted probability for the trajectory τ^i, $\hat{\tau}$ is the ground truth trajectory, λ and β are weighting factors, and $\mathcal{L}_{\text{reg}}$ is a regularization loss. Then, we minimize the following loss

$$\mathcal{L} = \mathcal{L}_{\text{IL}}^{i^*} + \mu \mathcal{L}_{\text{NLL}}^{i^*}, \quad \text{where} \;\; \mathcal{L}_{\text{NLL}}^{i^*} = -\log p_{i^*}, \tag{4}$$

that takes into account the selected trajectory τ^{i^*} and combines the imitation and the matching loss.

3.2 Inference-time planning policy

At inference time, we leverage the diverse predicted trajectories τ_i to compute the cost c_i of executing each of them. Then, we select the trajectory $\hat{i} = \arg\min_i c_i$ with minimum cost.

MinCost policy. Given the predicted SDV's trajectories and the associated probabilities, we can simply define the cost to be negatively proportional to the predicted probability: $c_i = -p_i$. However, this trivial approach ignores other road agents' future locations and thus may lead to collisions, as models can still predict colliding trajectories even when trained with auxiliary collision losses [3].

MinCostCC policy. In the work of [32], a safety layer is added to enforce safety constraints, e.g. collision avoidance, and legality constraints, e.g. respecting road rules, by checking the SDV trajectory predicted by the ML-based planner. In case of violations, a fallback trajectory generated by a rule-based planner is used instead of the ML-based one. While this approach was shown to substantially increase safety, the fallback trajectories are generated using a rule-based system that does not improve with data. As a result, the performance of the ML planner — which can improve by training on more data — is capped by the hand-engineered rule-based planner.

To overcome these limitations, we propose to improve the safety of the planner by leveraging the predicted SDV trajectory distribution and agents' predictions instead of relying on external modules. To this end, we first perform a Collision Check between each future SDV trajectory τ^i and the most probable predicted agents' locations ν^{j*} by means of overlap between their bounding boxes. We use the Separating Axis Theorem (SAT) [6] for efficient computation. Then, we extend the cost defined previously by adding a cost for any potential collision with other agents:

$$c_i = -p_i - \alpha \bar{t}_i \tag{5}$$

where α is a fixed penalty term and $\bar{t}_i$ is the timestep of the first predicted collision. In other words, we penalize SDV trajectories that are predicted to collide with road agents most probable futures. If the predicted set of trajectories contains at least one collision-free trajectory or trajectories with collisions further ahead in the predicted horizon, then our approach can improve the safety of the planner.

4 Experimental Evaluation

In this section, we first present the proprietary dataset we used to train and validate SafePathNet, and our experimental setting. Then, we present results of our evaluation, which include simulation, comparison of prediction performance on a public dataset, and public road tests. We also present an ablation study of our major components. While we are unable to release our proprietary simulator and dataset, we plan to release code and hyperparameters for training and testing our model on a public dataset to facilitate reproducibility.

4.1 Dataset and Experimental Setting

We use a proprietary dataset collected on an SDV platform in challenging urban areas of San Francisco and Palo Alto to train and evaluate our models. The dataset is composed of scenes spanning from 10 to 30 seconds and containing SDV recorded trajectory, HD map, and outputs of a proprietary perception system. Training and validation sets have 270 and 60 hours of driving respectively.

We trained the neural model on our training dataset applying the objectives presented in Section 3. We used the Adam optimizer and a base learning rate of 10^{-3} with cosine update schedule [18]. We apply synthetic perturbations to the training data [25, 32] to improve closed-loop performance. We trained for 40 epochs on a cluster with 8 nodes having 8 GPUs each, using a batch size of 64 samples per replica. Our model has 3 Transformer encoder and 3 decoder layers, with 5.7M parameters in total. Training takes about 5 hours. For more information, please refer to the Appendix, Section A.2.

Table 1: Comparison with competitors in closed-loop virtual evaluation. Results are reported in number of events per 1k miles with the .95 confidence interval. Lower is better for all metrics.

Planner type	Estimated Contacts	Close calls	Discomfort Brakings	Passiveness
ML planner [32]*	71 (52, 97)	116 (91, 148)	45 (31, 67)	85 (64, 113)
ML planner + fallback layer [32]*	42 (28, 63)	75 (55, 101)	284 (243, 331)	737 (670, 810)
SafePathNet (ours)	44 (29, 65)	87 (66, 116)	69 (50, 95)	133 (106, 167)

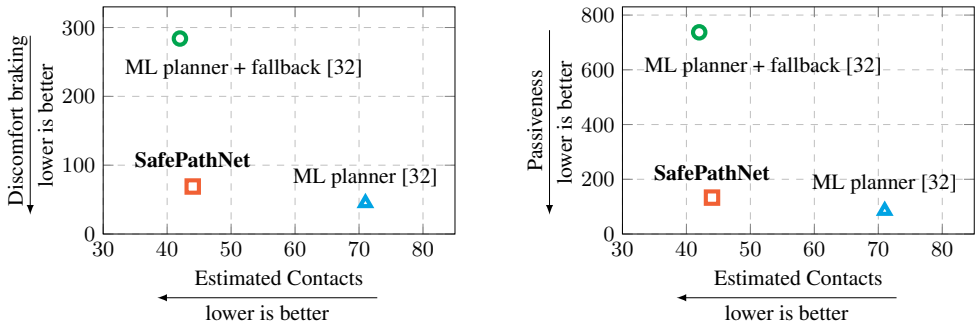

Figure 3: Comparison of Estimated Contacts against Discomfort Brakes/Passiveness for different planners, in events per 1k miles. Lower is better. Our model presents a better trade-off between comfort and safety.

4.2 Results from Simulation

We first evaluate our approach in simulation, in two different settings, relying on recorded scenes from the real world. To compare with other methods, we use a high-fidelity simulator which simulates the SDV physics through a kinematic model and log-replay road agents applying a longitudinal speed control to avoid front/side collisions. The SDV drives for approximately 580 miles. For ablation studies, we use an efficient simulator where we do not apply any kinematic constraints, even though the planner output is constrained by a kinematic model, and log-replay road agents. In this case, the SDV drives for about 340 miles. The simulators use a high-level representation of the scene; they do not simulate raw sensor data. In this Section, we compare with our implementation inspired by SafetyNet [32], since we do not have access to the original implementation.

The reported metrics are: (i) Estimated Contacts (ECs), i.e. number of times the SDV bounding box is closer to a road agent than 5cm or static obstacle by 1cm; (ii) Close Calls, i.e. number of times the SDV time-to-collision is less than 1.5 seconds, or SDV time headway is less than 1 second; (iii) Discomfort Brakes (DBs), i.e. number of excessive braking events; (iv) Passiveness, i.e. number of times SDV drives slower than in the log by at least 5 m/s while being behind the log. These metrics are computed as the number of events per 1k driven miles. We also report minADE and minFDE, i.e. minimum average and final distance error.

We first present results obtained with the high-fidelity simulation. As can be seen in Table 1, the naive ML-based planner experiences a high number of Estimated Contacts and Close calls to other road agents. We also report Discomfort Brakes (DBs) and Passiveness, showing the number of excessive braking events and the number of times SDV drives slower than in the log respectively, per 1k miles. Enabling the rule-based fallback layer on top of ML-based predictions [32], the safety of the planner is significantly improved, at the expense of passiveness and comfort (see large increase in DBs). On the contrary, our fully ML-based model presents comparable performance in terms of estimated contacts and close calls while having a limited impact to DBs and passiveness. This result confirms that our approach, leveraging the modelled SDV trajectory distribution and the agent predictions, improves the safety of the planner without relying on any external rule-based trajectory generator or impacting the riding comfort significantly (Fig. 3).

Qualitative samples comparing the two polices in our closed-loop simulator are shown in Figure 5. As highlighted by the green arrows, MinCostCC can successfully avoid collisions with other road agents. Figure 6 shows additional qualitative samples obtained using the MinCostCC policy. We find that SDV trajectories mostly differ in speed and acceleration profile due to route conditioning, but

Table 2: Comparison of prediction results on Lyft Motion Prediction Dataset.

Method	minADE@3s	minFDE@3s
SimNet [4][1]	0.71	1.38
Trajectron++ [27][20]	0.23	0.40
HAICU [14][20]	0.26	0.38
Ours[1]	0.43	0.86
Ours[20]	0.22	0.31

Table 3: Simulation ablation study comparing models trained with or without probability head and multiple experts (mean $\pm$ std over 5 runs).

Prob. head	Multiple experts		events per 1k miles		minADE@3s	
	SDV	agents	MinCost	MinCostCC	SDV	agents
✓	✓	✓	300 $\pm$ 17.3	183 $\pm$ 20.7	0.157 $\pm$ 0.004	0.738 $\pm$ 0.005
	✓	✓	571 $\pm$ 175	359 $\pm$ 35.4	0.081 $\pm$ 0.000	0.702 $\pm$ 0.002
✓	✓		297 $\pm$ 17.9	181 $\pm$ 23.1	0.156 $\pm$ 0.004	0.838 $\pm$ 0.007
✓		✓	298 $\pm$ 18.9	298 $\pm$ 18.9	0.316 $\pm$ 0.003	0.726 $\pm$ 0.007

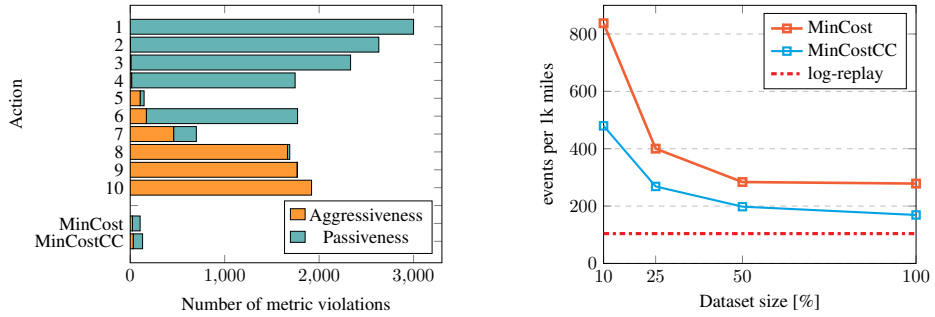

Figure 4: (a) We evaluate a single model using fixed expert selection policy. The model learns a diverse set of driving profiles, ranging from aggressive to passive. When augmenting MinCost policy with collision check (MinCostCC), we get a safe and comfortable driving policy. (b) We train our model using different subsets of training data, and evaluate them using MinCost and MinCostCC policies. MinCostCC policy requires fewer data samples to achieve performance of MinCost. Moreover, it keeps improving with more data, while MinCost performance plateaus.

show diverse curvature when turning and nudging parked vehicles. Agent trajectories vary in both curvature, speed and acceleration profile.

4.3 Results on Motion Prediction

We also compare the quality of our agent predictions with the literature on the public Lyft Motion Prediction Dataset [13]. Results in terms of minADE and minFDE are reported in Table 2. The superscript value on each method represents the number of per-agent predicted trajectories. As shown, our approach performs on par with or better than other methods and our unimodal baseline.

4.4 Results from Road Testing

Our model runs in real time on an SDV, integrated in the autonomous driving stack (10Hz). Thus, it can be deployed and tested on an SDV in the real world.

To validate our proposal in the real world, we first deployed and evaluated it on a private track. In particular, we successfully tested interactions with lead vehicles, including late braking at intersections and hard braking, and handling of traffic light intersections. Then, we tested SafePathNet on public roads letting it drive the SDV in full autonomy in Palo Alto. As can be seen in the attached video, our approach correctly handles challenging scenarios and safely drives on public roads. This includes complex situations and intersections where many other road agents are present. The road deployment confirmed that the planner was comfortable and felt safe for the SDV passengers.

4.5 Ablation study

In this section we present results of an empirical ablation study performed in our efficient simulation. The goal of the ablation study is to validate that the model learns a wide range of driving profiles, and that all parts of the model are required for safe and comfortable driving.

We report significant events per 1k miles (e.g. estimated contacts, close calls, divergence from route) and minADE metrics on the validation set. We run the planner in log replay mode to obtain baseline

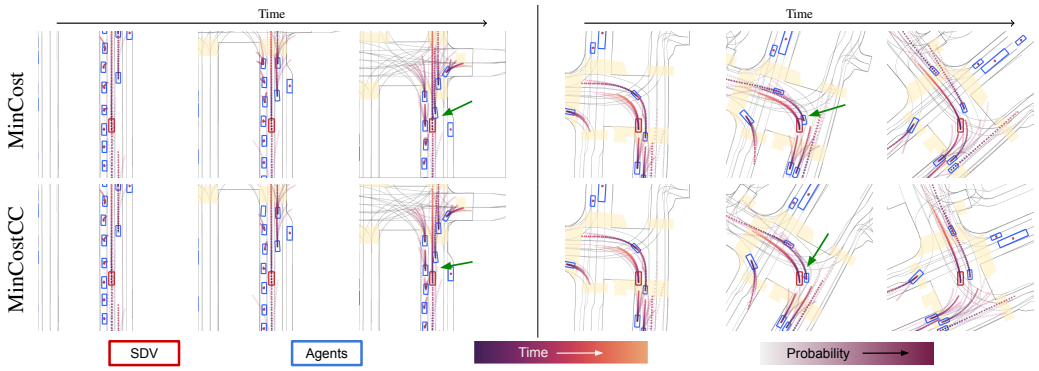

Figure 5: Qualitative results of SafePathNet, comparing MinCost and MinCostCC policy. Combining SafePathNet and MinCostCC improves the safety by reducing the estimated contacts with other road agents. (a) In the first simulated scene, we see the SDV colliding with the agent in the third shown frame when using MinCost whereas the SDV slows down using MinCostCC. (b) MinCost policy causes a collision with the turning agent in the second frame whereas MinCostCC shows safe distance between SDV and agent.

events per 1k miles. Since the dataset was created using an automated perception system, SDV and agent bounding boxes might overlap due to localization errors, resulting in baseline events per 1k miles close to 100.

Evaluating driving policies separately. In this study, we take a single model predicting 10 SDV experts and 5 agent experts, and perform simulation using fixed SDV expert selection, ignoring probabilities and collision checks. Results can be seen in Fig. 4(a). We show aggressiveness and passiveness metrics, indicating whether the simulated SDV advances or lags behind compared to logged position. As can be seen, SDV experts cover a wide range of behaviors, from fast aggressive to slow passive driving. Learning expert selection gives a safe and comfortable policy.

Planning policy. To validate that MinCostCC policy shows fewer estimated contacts than MinCost, and that the model improves with more training data, we train multiple models using 10%, 25%, 50% and 100% of training data, and evaluate using both policies. Results are in Fig. 4(b). Both policies improve with more data samples up to 50%, after which only MinCostCC improves. MinCostCC requires significantly fewer training samples to achieve similar performance and can steadily improve adding more training data.

Disabling probability distribution learning. We train a baseline model and a model without the probability head in Eq. (3) and (4), setting $\lambda_{\text{SDV}} = \mu_{\text{SDV}} = 0$. Results can be found in Table 3. The model without the probability head shows significantly lower SDV minADE, but shows poor driving policy with many estimated contacts, as expected.

Unimodal predictions. To validate the importance of having multiple experts for agent prediction and SDV planning, we train a model setting the number of agent trajectories M to 1 and another setting the number of SDV plans N to 1. Results can be found in Table 3. In the case of agent trajectories, setting $M = 1$ leads to higher agent minADE, confirming that using multiple agent trajectories is beneficial for the predictions, even though it does not significantly affect the events per 1k miles with collision check policy. In the case of SDV plans, setting $N = 1$ leads to substantially more events per 1k miles and higher SDV minADE, confirming that modelling multiple SDV trajectories is beneficial for both the policies.

5 Discussion

In this paper we presented SafePathNet, a unified neural model for prediction and planning that learns a distribution over future trajectories, corresponding to different driving profiles. Leveraging the predicted trajectory distribution, the safety of the planner is improved by penalizing SDV plans

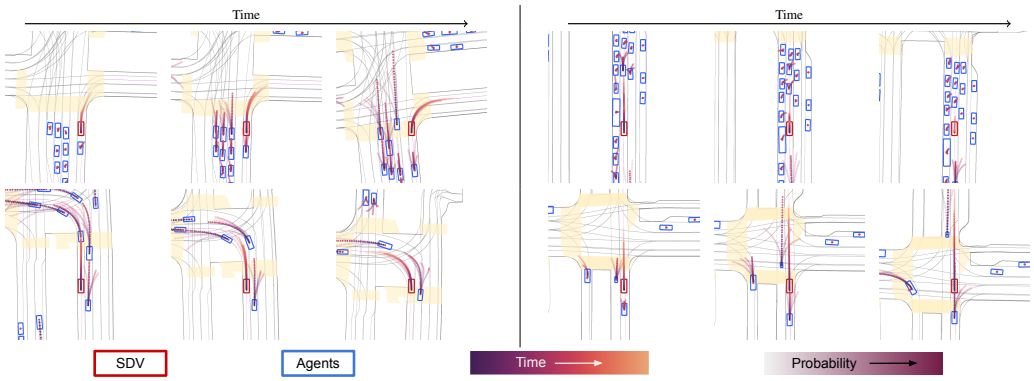

Figure 6: Additional qualitative results of SafePathNet. (i) In the top left sample, the SDV makes a right turn. The planned trajectory curves as expected at the intersection. (ii) The top right sample shows a scene with dense traffic. The SDV comfortably comes to a stop while the other agents around it are stopped. (iii) In the bottom left sample, the SDV approaches a left turn. The agents around it also have accurate predictions. (iv) In the bottom right sample, the SDV proceeds at a green traffic light. Different speed profiles can be seen. The agent to the left of SDV (in the leftmost lane) changes its most probable mode from going straight to left turn as it continues through the intersection.

that could lead to collisions with other road agents. We evaluated our approach through a realistic simulator and tested it on a real SDV in both our private testing facility and on public roads, showing that SafePathNet can be safely deployed in the real world.

Our method also has some limitations. For instance, the predicted distribution is not guaranteed to contain collision-free trajectories in all cases. However, predicted trajectories improve by adding diverse training data, increasing the chance of having a collision-free trajectory (see Fig. 4(b)). Moreover, even when a collision is expected, our model can increase the time-to-collision, increasing the chances of finding a collision-free trajectory in subsequent re-planning cycles, before the potential collision occurs. Also, our model does not guarantee scene consistency. Given the positive results in simulation and on the road, we speculate that the full scene consistency is not essential to improve the safety of our SDV planning approach. Another limitation is that, even though our method does not rely on a rule-based trajectory generator, we use a handcrafted cost for the trajectory selection. Still, the cost is used only for trajectory selection rather than trajectory generation. Finally, while we acknowledge that our closed-loop results are not reproducible due to the reliance on proprietary datasets and simulation tooling, we believe that our experiments clearly demonstrate the benefits of our approach, not only in our closed-loop and on-road tests, but also via competitive prediction results on the Lyft Motion Prediction dataset.

Acknowledgments

We thank the whole Woven Planet, and in particular the research, motion planning, SimOps, TechOps and release teams, for support and discussions. We also thank Nicolas Carion and Yann LeCun for discussions.

References

[1] A. Amini, G. Rosman, S. Karaman, and D. Rus. Variational end-to-end navigation and localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8958–8964. IEEE, 2019.

[2] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[3] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worsts. In *Proceedings of Robotics: Science and Systems XV*, 2019.

[4] L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osinski, H. Grimmett, and P. Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars, 2016.

[6] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[8] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision*, pages 624–641. Springer, 2020.

[9] A. Cui, S. Casas, A. Sadat, R. Liao, and R. Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16107–16116, 2021.

[10] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[11] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. VectorNet: Encoding HD maps and agent dynamics from vectorized representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[12] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, and A. Kendall. Urban driving with conditional imitation learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257, 2020.

[13] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. *arXiv preprint arXiv:2006.14480*, 2020.

[14] B. Ivanovic, K.-H. Lee, P. Tokmakov, B. Wulfe, R. McAllister, A. Gaidon, and M. Pavone. Heterogeneous-agent trajectory forecasting incorporating class uncertainty. *arXiv preprint arXiv:2104.12446*, 2021.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[16] A. Kamenev, L. Wang, O. B. Bohan, I. Kulkarni, B. Kartal, A. Molchanov, S. Birchfield, D. Nistér, and N. Smolyanskiy. Predictionnet: Real-time joint probabilistic traffic prediction for planning, control, and simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8936–8942. IEEE, 2022.

[17] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. In *International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[18] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*, 2017.

[19] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[20] C. Paduraru, D. J. Mankowitz, G. Dulac-Arnold, J. Li, N. Levine, S. Gowal, and T. Hester. Challenges of real-world reinforcement learning:definitions, benchmarks & analysis. *Machine Learning Journal*, 2021.

[21] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14062–14071, 2020.

[22] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, 1989.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[24] M. Riedmiller, M. Montemerlo, and H. Dahlkamp. Learning to drive a real car in 20 minutes. In *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, pages 645–650. IEEE, 2007.

[25] S. Ross, G. J. Gordon, and J. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[26] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.

[27] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

[28] S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[29] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[30] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821. IEEE, 2022.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[32] M. Vitelli, Y.-X. Chang, Y. Ye, A. Ferreira, M. Wołczyk, B. Osi'nski, M. Niendorf, H. Grimmett, Q. Huang, A. Jain, and P. Ondruska. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. *ArXiv*, abs/2109.13602, 2021.

[33] M. Werling, S. Kammel, J. Ziegler, and L. Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, 2012.

[34] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[35] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen. Imitation learning: Progress, taxonomies and opportunities. *arXiv preprint arXiv:2106.12177*, 2021.

A Appendix

A.1 On-road evaluation (attached video)

In the supplementary video, available on the project webpage at `https://woven.mobi/safepathnet`, we present the SDV driving **in full autonomy mode** in the streets of Palo Alto (CA), in the following scenarios:

- **Yielding to cyclists at a stop intersection**: SDV stops at an unregulated stop intersection, waits for cyclists to pass and safely proceeds.
- **Left turn and passing a parked unloading truck**: SDV stops at a regulated intersection following a lead vehicle and then executes left turn while also nudging a parked unloading truck.
- **Cut-in maneuver**: A vehicle merges into SDV lane from an adjacent lane. SDV comfortably slows down.
- **Stop and left turn**: The SDV comes to a safe stop at the intersection before proceeding to make a left turn
- **Urban driving in dense traffic**: Multiple stretches of dense urban driving in full autonomy mode without any safety driver interventions.

The above on-road test was executed without using rule-based planners. Note that the video shows agent trajectory predictions from a standalone integrated system for visualization purposes, which we plan to update with SafePathNet predictions. Our approach correctly handles challenging scenarios and safely drives on public roads. This includes complex situations and intersections where many other road agents are present. The road deployment confirmed that the planner was comfortable and felt safe for the SDV passengers.

A.2 Implementation Details

Architectural details

As input to the model, we use $K_a = 1$ second of agent history, while we use $K_s = 3$ seconds of SDV history represented by moving/stationary information. The model predicts $N = 10$ SDV trajectories with $T_s = 45$ steps and $M = 5$ agent trajectories with $T_a = 30$ steps, both at 10 Hz.

We use 3 layers of PointNet for encoding map and road agents polylines with 128 dimensions. The outputs are projected to $256d$ vectors and fed to the Transformer. The learnable query embeddings for SDV and road agents have the same size. We use $1024d$ feed-forward networks (FFN) inside the Transformer. Both Transformer encoder and decoder have 3 layers. The FFNs used as trajectory decoder are composed of a hidden $512d$ layer and an output layer with shape dependent of the prediction horizon. In total the network has 5.7M parameters.

Loss weights

Here we recap the matching cost and loss equations from the main draft.

We compute matching cost for each trajectory then select one trajectory according to

$$i^* = \arg\min_i \ \mathcal{L}^i_{\text{IL}} + \lambda(1 - p_i), \quad \text{where} \ \ \mathcal{L}^i_{\text{IL}} = \sum_{t=1}^{T} ||\tau^i_t - \hat{\tau}_t||_1 + \beta \mathcal{L}^i_{\text{reg}}, \tag{6}$$

where p_i is the predicted probability for the trajectory τ^i, $\hat{\tau}$ is the ground truth trajectory, λ and β are weighting factors, and $\mathcal{L}_{\text{reg}}$ is a regularization loss. λ controls the weight of the assignment probability compared to the imitation loss, while β controls the smoothness of the predicted trajectories. We empirically set $\lambda_{\text{SDV}} = 1$, $\beta_{\text{SDV}} = 1$, $\lambda_{\text{agent}} = 0.04$, $\beta_{\text{agent}} = 0$. We note that we do not apply regularization to the agent trajectories since they do not have to be executed. We also scale λ_{agent} taking into account that the agent outputs are not regularized and passed through a kinematic decoder.

We minimize the following loss

$$\mathcal{L} = \mathcal{L}^{i^*}_{\text{IL}} + \mu \mathcal{L}^{i^*}_{\text{NLL}}, \quad \text{where} \ \ \mathcal{L}^{i^*}_{\text{NLL}} = -\log p_{i^*}, \tag{7}$$

that takes into account the selected trajectory τ^{i^*} and combines the imitation and the matching loss. Similarly to DETR, we set $\mu = \lambda$ for both SDV and agents.

Overall, our total loss combines both SDV and agents losses, optimized simultaneously:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{SDV}} + \alpha \mathcal{L}_{\text{agent}}, \tag{8}$$

where α is agent loss coefficient. It is set to 10 in all experiments.

For perturbations we use the same hyperparameters as SafetyNet [32].