Vehicle trajectory prediction works, but not everywhere

Mohammadhossein Bahari¹,* Saeed Saadatnejad¹,* Ahmad Rahimi²,† Mohammad Shaverdikondori²,† Mohammad Shahidzadeh² Seyed-Mohsen Moosavi-Dezfooli³ Alexandre Alahi¹

¹EPFL  ²Sharif university of technology  ³ETH Zurich
{mohammadhossein.bahari, saeed.saadatnejad}@epfl.ch

Abstract

Vehicle trajectory prediction is nowadays a fundamental pillar of self-driving cars. Both the industry and research communities have acknowledged the need for such a pillar by running public benchmarks. While state-of-the-art methods are impressive, i.e., they have no off-road prediction, their generalization to cities outside of the benchmark is unknown. In this work, we show that those methods do not generalize to new scenes. We present a novel method that automatically generates realistic scenes that cause state-of-the-art models to go off-road. We frame the problem through the lens of adversarial scene generation. We promote a simple yet effective generative model based on atomic scene generation functions along with physical constraints. Our experiments show that more than 60% of the existing scenes from the current benchmarks can be modified in a way to make prediction methods fail (predicting off-road). We further show that (i) the generated scenes are realistic since they do exist in the real world, and (ii) can be used to make existing models robust by 30-40%. Code is available at https://s-attack.github.io/.

1. Introduction

Vehicle trajectory prediction is one of the main building blocks of a self-driving car, which forecasts how the future might unroll based on the scene, i.e., the road structure, and the traffic participants. State-of-the-art models are commonly trained and evaluated on datasets collected from few cities [14, 19, 23]. While their evaluation has shown impressive performance, i.e., almost no off-road prediction, their generalization to other types of possible scenes e.g., other cities, remains unknown. Figure 1 shows a real-world example where a state-of-the-art model reaching zero off-road in the known benchmark [19] fails in South St, New York, US. Since collecting and annotating data from all real-world scenes is not a viable and affordable solution, we present a method that automatically investigates the robustness of vehicle trajectory prediction to the scene. We tackle the problem through the lens of realistic adversarial scene generation.

Given an observed scene, we want to generate a realistic modification of the scene where prediction models fail. Having an off-road prediction is a clear indication of a failure in the scene reasoning of the model and was used in some previous works [8, 16, 36, 38]. To find a realistic example where the models go off-road, the huge space of possible scenes should be explored. A solution is learning generative models that mimic the distribution of a dataset. Yet, they will represent a portion of possible real-world scenes as they cannot generate scenes beyond what they have observed in the dataset. Hence, we suggest a simple yet efficient alternative. We show that we can use a limited num-

* Equal contribution as the first authors.
† Equal contribution as the second authors.
ber of simple functions for transforming the scene into new realistic but challenging ones. Our method can explicitly extrapolate to new scenes.

We introduce atomic scene generation functions where given a scene in the dataset, the functions generate multiple new ones. The functions are chosen such that they can cover a range of realistic scenes. Then, we choose the scenes where the prediction model produces an off-road trajectory. Using three state-of-the-art trajectory prediction models trained on Argoverse public dataset [19], we demonstrate that more than 60% of the existing scenes in the dataset can be modified in a way that it will make state-of-the-art methods fail (i.e., predict off-road). We confirm that the generated scenes are realistic by finding real-world locations that partially resemble the generated scenes. Also, we demonstrate off-road predictions of the models in those locations. To this end, we extract appropriate features from each scene and use image retrieval techniques to search over public maps [2]. We finally show that these generated scenes can be used to improve the robustness of the models.

Our contributions are fourfold: (1) We highlight the need for a more in-depth evaluation of the robustness of vehicle trajectory prediction models. (2) We propose an evaluation framework through the lens of realistic adversarial scene generation. We promote a simple yet effective generative model based on atomic scene generation functions. (3) We demonstrate that our generated scenes are realistic by finding similar real-world locations where the models fail. (4) We show that we can leverage our generated scenes to make the models more robust.

2. Related work

Vehicle trajectory prediction. The scene plays an important role in vehicle trajectory prediction as it constrains the future positions of the agents. Therefore, modeling the scene is common in spite of some human trajectory prediction models [13, 39]. In order to reason over the scene in the predictions, some suggested using a semantic segmented map to build circular distributions and outputting the most probable regions [21]. Another solution is reasoning over raw scene images using convolutional neural networks (CNN) [30]. Many follow-up works represented scenes in the segmented image format and used the learning capability of CNNs over images to account for the scene [10, 17, 18, 25, 40]. Carnet [44] used attention mechanism to determine the scene regions that were attended more, leading to an interpretable solution. Some recent work showed that scene can be represented by vector format instead of images [7, 24, 32, 46]. To further improve the reasoning of the model and generate predictions admissible with respect to the scene, use of symmetric cross-entropy loss [38], off-road loss [8], and REINFORCE loss [16] have been proposed. Despite all these efforts, there have been limited attention to assess the performance of trajectory prediction models on new scenes. Our work proposes a framework for such assessments.

Evaluating self-driving systems. Self-driving cars deal with dynamic agents nearby and the static environment around. Several works studied the robustness of self-driving car modules with respect to the status of dynamic agents on the road, e.g., other vehicles. Some previous works change the behavior of other agents in the road to act as attackers and evaluate the model’s performance with regards to the interaction with other agents [3, 4, 20, 26, 28, 42, 51]. Others directly modify the raw sensory inputs to change the status of the agents in an adversarial way [15, 48, 50, 52].

In addition to the dynamic agents, driving is highly dependent on the static scene around the vehicle. The scene understanding of the models can be assessed by modifying the input scene. Previous works modify the raw sensory input by changing weather conditions [33, 49, 53], generating adversarial drive-by billboards [29, 54], and adding carefully crafted patches/lines to the road [12, 45]. These works have not changed the shape of the scene, i.e., the structure of the road. In contrast, we propose a conditional scene generation method to assess the scene reasoning capability of trajectory prediction models. Also our approach is different from scene generation based on graph [35] or semantic maps [43]. Ours is an adversarial one which can extrapolate to new scenes.

3. Realistic scene generation

In this section, we explain in detail our approach for generating realistic scenes. After introducing the notations in Section 3.1, we show how we generate each scene in Section 3.2 and satisfy physical constraints in Section 3.3.

3.1. Problem setup

The vehicle trajectory prediction task is usually defined as predicting the future trajectory of a vehicle $z$ given its observation trajectory $h$, status of surrounding vehicles $a$, and scene $S$. For the sake of brevity, we assume $S$ is in the vector representation format [19]. Specifically, $S$ is a matrix of stacked $2d$ coordinates of all lanes’ points in $x$-$y$ coordinate space where each row represents a point $s = (s_x, s_y)$. Formally, the output trajectory $z$ of the predictor $g$ is:

$$z = g(h, S, a).$$  \hspace{1cm} (1)

Given a scene $S$, our goal is to create challenging realistic scene $S^*$ as we will explain in Section 3.2.

We show in Appendix C that our method is seamlessly applicable when $S$ is in image representation.
3.2. Conditional scene generation

Our controllable scene generation method generates diverse scenes conditioned on existing scenes. Specifically, our method consists of three types of parametric turns, which are controllable and can represent a range of difficult road shapes. To this end, we normalize the scene (i.e., translation and rotation with respect to \( h \)) apply the transformation functions, and denormalize to return the generated scene to the original view. Note that every transformation of \( S \) is followed by the same transformations on \( h \) and \( a \).

We define transformations on each scene point in the following form:

\[
\tilde{s} = (s_x, s_y + f(s_x - b))
\]

(2)

where \( f \) is a single-variable transformation function and \( b \) is the border parameter that determines the region of applying the transformation. In other words, we define \( f(< 0) = 0 \) so the areas where \( s_x < b \) are not changed. This confines the changes to the regions containing the prediction. One example is shown in Figure 2 in which \( b \) is set to 5 meters. The new scene is named \( \tilde{S} \), a matrix of stacked \( \tilde{s} \). We propose three classes of functions for the choice of \( f \).

**Smooth-turn:** this function represents different types of possible single turns in the road.

\[
f_{st, \alpha}(s_x) = \begin{cases} 0, & s_x < 0 \\ q_\alpha(s_x), & 0 \leq s_x \leq \alpha_1 \\ (s_x - \alpha_1)q_\prime_\alpha(\alpha_1) + q_\alpha(\alpha_1), & \alpha_1 < s_x \end{cases} \quad \alpha = (\alpha_1, \alpha_2, \alpha_3)
\]

(3)

where \( \alpha_1 \) determines the length of the turn, \( \alpha_2, \alpha_3 \) make the sharpness of it, and \( q_\alpha \) indicates the derivative of the defined auxiliary function \( q_\alpha \). Note that according to the definition, \( f_{st, \alpha} \) is continuously differentiable and makes a smooth turn. One such turn is depicted in Figure 2b.

**Double-turn:** these functions represent two consecutive turns with opposite directions. Also, there is a variable indicating the distance between them:

\[
f_{dt, \beta}(s_x) = f_{st, \beta_1}(s_x - \beta_2),
\]

(4)

\[
\beta = (\beta_{11}, \beta_{12}, \beta_{13}), \quad \beta_1 = (\beta_{11}, \beta_{12}, \beta_{13}),
\]

where \( \beta_1 \) is the set of parameters of each turn described in Equation (3) and \( \beta_2 \) is the distance between two turns. One example is shown in Figure 2c.

**Ripple-road:** one type of scene that can be challenging for the prediction model is ripple road:

\[
f_{rr, \gamma}(s_x) = \begin{cases} 0, & s_x < 0 \\ \gamma_1(1 - \cos(2\pi \gamma_2 s_x)), & s_x \geq 0 \end{cases}, \quad \gamma = (\gamma_1, \gamma_2)
\]

(5)

where \( \gamma_1 \) determines the turn curveatures and \( \gamma_2 \) determines the sharpness of the turns. One such turn is depicted in Figure 2d.

3.3. Physical constraints

Every scenario consists of a scene and vehicle trajectories in it. The generated scenarios must be feasible, otherwise, they cannot represent probable real-world cases. We consider a scenario as feasible if a human driver can pass it safely. This means that the physical constraints – i.e., the Newton’s law should not be violated. The Newton’s law indicates a maximum feasible speed for each road based on its curvature [22]:

\[
v_{max} = \sqrt{\mu g R},
\]

(6)

where \( R \) is the radius of the road, \( \mu \) is the friction coefficient, and \( g \) is the gravity. To consider the most conservative situation, we pick the maximum curvature (minimum radius) existing in the generated road. Then, we slow down the history trajectory when the speed is higher than the maximum feasible speed, and we name it \( \tilde{h} \). Note that this conservative speed scaling ensures a feasible acceleration too.

We will show in Section 5 that a model with hard-coded physical constraints successfully predicts the future trajectory for the generated scenes, which indicates that our constraints are enough.

4. Search method

In the previous section, we defined a realistic controllable scene generation method. Now, we introduce a search method to find a challenging scene specific to each trajectory prediction model.

4.1. Scene search method

We define \( m \) as a function of \( z \) and \( S \) measuring the percentage of prediction points that are off-road derived by a binary mask of drivable area. We aim to solve the following problem to obtain a scene for the prediction model that fails in:

\[
S^* = \arg \min_{\tilde{S}} l(\tilde{z}, \tilde{S}),
\]

(7)

\[
l(\tilde{z}, \tilde{S}) = \left(1 - m(\tilde{z}, \tilde{S})\right)^2
\]

where \( \tilde{S} \) is a modification of \( S \) according to Equation (2).
Equation (4), or Equation (5). Moreover, \( \tilde{z} = g(h, \hat{S}, \tilde{a}) \) is the model’s predicted trajectory for the modified scene. The optimization problem finds the corresponding parameters to obtain \( S^* \) that gives the highest number of off-road prediction points. Equation (7) can be optimized using any black-box optimization technique. We have studied Bayesian optimization [41, 47], Genetic algorithms [5, 34], Tree-structured Parzen Estimator Approach (TPE) [9] and brute-force.

4.2. Overall algorithm

In this section, we demonstrate the overall algorithm employing the previously explained components. The pseudo-code of the algorithm for generating a scene is shown in Algorithm 1. The goal is to generate the scene \( S^* \) for a given scenario \( x, a, S \) and predictor \( g \). The process is called for \( k_{\text{max}} \) iterations. In each iteration, we start with selecting a transformation function (L. 3). Then, the transformation function generates the corresponding scene (L. 4). After that, the observation trajectory is scaled to ensure feasibility of the scenario (L. 5). Next, the prediction of the model in the new scenario is computed and used to calculate the loss (L. 6,7). The best-achieved loss determines the final generated scene.

5. Experiments

We conduct experiments to answer the following questions: (1) How is the performance of the prediction models on our generated scenes? (2) Are the generated scenes realistic and possibly similar to the real-world scenes? (3) Are we able to leverage the generated scenes to improve the robustness of the model?

5.1. Experimental setup

5.1.1 Baselines and datasets

We conduct our experiments on the following baselines: LaneGCN [31]. It constructs a lane graph from vectorized scene and uses self-attention to learn the predictions.

This method was among the top methods in Argoverse Forecasting Challenge 2020 [1]. It is a multi-modal prediction model which also provides the probability of each mode. Therefore, in our experiments, we consider the mode with the highest probability.

DATF [38]. It is a flow-based method which uses a symmetric cross-entropy loss to encourage producing on-road predictions. This multi-modal prediction model does not provide the probability of each mode. We therefore consider the mode which is closest to the ground truth.

WIMP [27]. They employ a scene attention module and a dynamic interaction graph to capture geometric and social relationships. Since they do not provide probabilities for each mode of their multi-modal predictions, we consider the one which is closest to the ground truth.

MPC [6, 55]. We report the performance of model predictive control (MPC) as a rule-based model which tries to follow the road while satisfying the kinematic constraints. The kinematic constraints are chosen based on [6].
our baselines were trained on. Given the 2 seconds observation trajectory, the goal is to predict the next 3 seconds as the future motion of the vehicle. It is a large scale vehicle trajectory dataset. The dataset covers parts of Pittsburgh and Miami with total size of 290 kilometers of lanes.

5.1.2 Metrics

Hard Off-road Rate (HOR): in order to measure the percentage of samples with an inadmissible prediction with regards to the scene, we define HOR as the percentage of scenarios that at least one off-road happens in the prediction trajectory points. It is rounded to the nearest integer.

Soft Off-road Rate (SOR): to measure the performance in each scenario more thoroughly, we measure the percentage of off-road prediction points over all prediction points and the average over all scenarios is reported. The reported values are rounded to the nearest integer.

5.1.3 Implementation details

We set \( k_{\text{max}} \) to 60, the friction coefficient \( \mu \) to 0.7 [11] and \( b \) equal to 5 for all experiments. We developed our model using a 32GB V100 NVIDIA GPU. For the choice of the black-box algorithm, as the search space of parameters is small in our case, we opt for the brute-force algorithm. We will report the performance of other black-box methods in Appendix B.2.

5.2. Results

We first provide the quantitative results of applying our method to the baselines in Table 1. The last column represents the results of the search method described in Section 3.3. We also reported the performance of considering only one category of scene generation functions in the optimization problem Equation (7). The results indicate a substantial increase in SOR and HOR across all baselines in different categories of the generated scenes. This shows that the generated scenes are difficult for the models to handle. LaneGCN and WIMP have competitive performances, but WIMP run-time is 50 times slower than LaneGCN. Hence, we use LaneGCN to conduct our remaining experiments.

Figure 3 visualizes the performance of the baselines in our generated scenes. We observe that all models are challenged with the generated scenes. More cases are provided in the Appendix A.

In Table 1, we observe that SOR is less than or equal to 1% for all methods in the original scenes. Our exploration shows that more than 90% of these off-road cases are due to the annotation noise in the drivable area maps of the dataset and the models are almost error-free with respect to the scene. Some figures are provided in Appendix A. While this might lead to the conclusion that the models are flawless, results on the generated scenes question this conclusion. We confirm our claim in the next section by retrieving the real-world scenes where the model fails.

Feasibility of a scenario is an important feature for generated scenes. As mentioned in Section 3.3, we added physical constraints to guarantee the physical feasibility of the scenes. Table 1 indicates that MPC as a rule-based model predicts almost without any off-road in the generated scenarios. It approves that the scenes are feasible with the given history trajectory. In order to study the importance of added constraints, we relax the constraints for the generated scenes. We report the performance of the baseline and MPC on the cases where the maximum speed in their \( v \) is higher than \( v_{\text{max}} \). In Table 2, we observe that without those feasibility-assurance constraints, there are more cases where MPC is unable to follow the road and has \( 3 \times \) more off-road. We conclude that those constraints are necessary to make the scene feasible. We keep the constraints in all of our experiments to generate feasible scenarios.

5.3. Real-world retrieval

So far, we have shown that the models fail in some generated scenes. This would confirm the usefulness of the proposed framework in finding realistic challenging scenes. Inspired by image retrieval methods [37], we develop a retrieval method to find similar roads in the real-world. First, we extract data of 4 arbitrary cities (New York, Paris, Hong Kong, and New Mexico) using OSM [2]. Then, 20,000 random samples of \( 200 \times 200 \) meters are collected from each city. Note that it is the same view size as in Argoverse samples. Then, a feature extractor is required to obtain a feature vector for each scene. We used the scene feature extractor of LaneGCN named MapNet to obtain some 128 dimensional feature vectors for each sample. We then use the well-known image retrieval method K-tree algorithm [37]. It first uses K-Means algorithm multiple times to cluster the feature vectors of all scenes into a predefined number of clusters (in our case 1000 as 1% representative of different scenes). Then, given a generated scene as the query, it sorts real scenes based on the similarity with the query scene and retrieves 10 closest scenes to the query. Finally, we test the prediction model in these examples. Some examples are provided in Figure 4. More scenes can be found in Appendix A.

5.4. Robustness

Here, we study if we can make the models robust against new generated scenes. To this end, we fine-tune the trained model using a combination of the original training data and the generated examples by our method for 10 epochs.

We report the performance of these models in the generated scenes with different transformation power. Transformation power is determined by \( \alpha_2 \times 3000 \), \( \beta_1 \times 3000 \) and \( \gamma_1 \) for Equation (3), Equation (4), and Equation (5), respec-
Table 1. Comparing the performance of different baselines in the original dataset scenes and our generated scenes. SOR and HOR are reported in percent and the lower represent a better reasoning on the scenes by the model. MPC as a rule-based model always has on-road predictions both in original and our generated scenes.

Table 2. Impact of the physical constraints. We report the performance with and without the physical constraints explained in Section 3.3. The numbers are reported on samples of data with speed higher than $v_{max}$ in their $h$.

Table 3 indicates that without losing the performance in the original accuracy metrics, the fine-tuned model is less vulnerable to the generated scenes by predicting 40% less SOR and 30% less HOR in the Full setting. While the results show improvements in all transformation powers, the gains in extreme cases are higher, i.e., the model can handle them better after fine-tuning.

In Figure 5, the prediction of the original model is compared with the prediction of the robust model. The original model cannot predict without off-road while the fine-tuned model is able to predict reasonable and without any off-road point.

5.5. Discussions

In this section, we perform experiments and bring speculations to shed light on the weaknesses of the models.

1) We study the ability to transfer the generated scenes to new models, i.e., how models perform on the scenes generated for other models. We conduct this experiment by storing the generated scenes for a source model which lead to an off-road prediction, and evaluate the performance of target models on the stored scenes. Table 4 shows that the transferred scenes are still difficult cases for other models.

2) We study how models perform with smoothly changing the transformation functions parameters. To this end, we smoothly change the transformation parameters for 100 random scenes and visualize the heatmap of HOR for the generated scenes. Figure 6 demonstrates that models are more vulnerable to larger transformation parameters, i.e., sharper turns. Also, it shows more off-road in the left turns compared with the right ones. This could be due to the biases in the dataset [36]. A clear improvement is visible in the robust model.

3) Our experiments showed that while the model has almost zero off-road rate in the original scenes, it suffers from over 60% off-road rate in the generated ones. In order to
Figure 4. Retrieving some real-world locations similar to the generated scenes using our real-world retrieval algorithm. We observe that the model fails in Paris (a), Hong Kong (b) and New Mexico (c).

| Model               | Pow=1 SOR/HOR | Pow=3 SOR/HOR | Pow=5 SOR/HOR | Pow=7 SOR/HOR | Pow=9 (Full) SOR/HOR |
|---------------------|---------------|---------------|---------------|---------------|----------------------|
| LaneGCN             | 2 / 8         | 12 / 35       | 19 / 49       | 22 / 58       | 23 / 66              |
| LaneGCN w/ aug      | 1 / 7         | 6 / 21        | 10 / 30       | 13 / 38       | 14 / 46              |

Table 3. Comparing the original model and the fine-tuned model with data augmentation of the generated scenes. The performance is reported on generated scenes with different transformation power (Pow). Transformation power is determined by $\alpha \times 3,000$, $\beta \times 3,000$ and $\gamma$ for Equation (3), Equation (4), and Equation (5), respectively which represents the amount of curvature in the scene. The average / final displacement errors on original scenes are equal to 1.35/2.98m for both original and fine-tuned models.

Figure 5. The output of the original model (the left) vs the robust model (the right) in a generated scene. While the original model has a trajectory in non-drivable area, the robust model predicts without any off-road.

hypothesize the causes of this gap, we explored the training data. We observed that in most samples, the history $h$ has enough information about the future trajectory which reduces the need for the scene reasoning. However, our scene generation approach changes the scene such that $h$ includes almost no information about the future trajectory. This essentially makes a situation which requires scene reasoning. We speculate that this feature is one factor which makes the generated scenes challenging. Note that this does not contradict with the ablations in [31] as their performance measure is accuracy. Figure 7a shows a failure of the model where the prediction is only based on $h$ instead of reasoning over the scene. However, the robust model learned to reason over the scene, as shown in Figure 7b. While our discussion is an observational hypothesis, we leave further studies for future works.

4) In some cases, our generated scene could not lead to an off-road prediction, as seen in the numerical results. One example is depicted in Figure 8a.
5.6. Limitations

While our method offers a new approach for assessing trajectory prediction models, it has some limitations. First, our transformation functions are limited, and they cannot cover all real-world cases. Second, in addition to the off-road criterion, there exist other failure criteria. For instance, collision with other agents or abnormal behaviors like sudden lane changes. Figure 8b shows one scenario in which the model prediction is in the drivable area but the sudden lane change is abnormal.

6. Conclusion

In this work, we presented a conditional scene generation method. We showed that several state-of-the-art trajectory prediction models fail in our generated scenes. Notably, they have high off-road rate in their predictions. Next, leveraging image retrieval techniques, we retrieved real-world locations which partially resemble the generated scenes and demonstrate their failure in those locations. We made the model robust against the generated scenes. We hope that this framework helps to better evaluate the prediction models which are involved in the autonomous driving systems.

Potential negative societal impact: Self-driving cars...
have many advantages including reducing carbon footprint and less traffic on roads. However, still the main challenge is their safety. People have concerns about it and wonder machines’ behaviors in new unseen scenarios. We believe more research in assessing the models is needed to provide trust.

7. Acknowledgments

This project was funded by Honda R&D Co., Ltd and from the European Union’s Horizon 2020 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 754354.

References

[1] Argoai challenge 2020. CVPR Workshop on Autonomous Driving. 4
[2] Openstreetmap. https://www.openstreetmap.org. 2, 5
[3] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In International Conference on Robotics and Automation (ICRA). IEEE, 2019. 2
[4] Matthias Althoff and Sebastian Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In IEEE Intelligent Vehicles Symposium (IV), 2018. 2
[5] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Man B. Srivastava. Genattack: practical black-box attacks with gradient-free optimization. GECCO, 2019. 4
[6] Mohammadhossein Bahari, Ismail Nejjar, and Alexandre Alahi. Injecting knowledge in data-driven vehicle trajectory predictors. Transportation Research Part C: Emerging Technologies, 128, 2021. 4
[7] Mohammadhossein Bahari, Vahid Zehtab, Sadegh Khosrasani, Sana Ayramly, Saeed Saadatnejad, and Alexandre Alahi. Svg-net: An svg-based trajectory prediction model. arXiv preprint arXiv:2110.03706, 2021. 2
[8] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. Robotics: Science and Systems (RSS), 2019. 1, 2
[9] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems (NeurIPS), 2011. 4, 14
[10] Yuriy Biktairov, Maxim Stebelev, Irina Rudenko, Oleh Shliakhko, and Boris Yangel. Prank: motion prediction based on ranking. Advances in Neural Information Processing Systems (NeurIPS), 2020. 2
[11] Peter J. Blau. Friction Science and Technology: From Concepts to Applications. 2005. 5
[12] Adith Boloor, Karthik Garimella, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Attacking vision-based perception in end-to-end autonomous driving models. Journal of Systems Architecture, 110, 2020. 2
[13] Smail Ait Bouhssain, Saeed Saadatnejad, and Alexandre Alahi. Pedestrian intention prediction: A multi-task perspective. European Association for Research in Transportation (hEART), 2020. 2
[14] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 1
[15] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. arXiv preprint arXiv:1907.05418, 2019. 2
[16] Sergio Casas, Cole Gulino, Simon Suo, and Raquel Urtasun. The importance of prior knowledge in precise multimodal prediction, 2020. 1, 2
[17] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In Conference on Robot Learning. PMLR, 2018. 2
[18] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. Conference on Robot Learning; 2019. 2
[19] 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 (CVPR), 2019. 1, 2, 4
[20] Anthony Corso, Robert J Moss, Mark Koren, Ritchie Lee, and Mykel J Kochenderfer. A survey of algorithms for black-box safety validation. arXiv preprint arXiv:2005.02979, 2020. 2
[21] Pasquale Coscia, Francesco Castaldo, Francesco A.N. Palmieri, Alexandre Alahi, Silvio Savarese, and Lamberto Ballan. Long-term path prediction in urban scenarios using circular distributions. Image and Vision Computing, 69, 2018. 2
[22] Halliday David, Robert Resnick, and Earl Walker. Fundamentals of physics. Wiley, 1997. 3
[23] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabekh Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. 2021. 1
[24] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 2
[25] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 2
A. Additional qualitative results

1. **Real-world retrieval images.** We show more real-world examples for both cases where the trajectory prediction model fails and succeeds in Figure 9.

2. **More generated scenes.** Figure 10 provides more visualizations for the performance of the baselines in our generated scenes.

3. **Noise in the drivable area map.** The models predict near perfect in the original dataset with HOR of less than 1%. Our exploration shows that most of the 1% failed cases are due to the annotation noise in the drivable area maps of the dataset and the models are almost error-free with respect to the scene. Some figures are provided in Figure 11.

4. **Gifs.** We provide gifs on the performance of the model when smoothly transforming the scene in Figure 12. We observe that in some cases the model fails and in some succeeds.

B. Additional quantitative results

B.1. **Excluding trivial scenes.**

In this section, we remove some trivial scenes, i.e., the scenes that fooling is near impossible, e.g., the scenes with zero velocity. Excluding them, we report in Table 5 and compared to Table 1 of the paper, the off-road numbers substantially increase.

B.2. **Exploring black box algorithms.**

In the paper, we mentioned that we used a brute-force approach for finding the optimal values as the search space is not huge. Here, we investigate different black box algorithms for the search. The results of applying different search algorithms are provided in Table 6. They cannot overcome the brute-force approach because of their bigger search spaces (the continuous space instead of the discrete space) and the large required computation time.

C. **Generalization to rasterized scene.**

In the paper, we assumed $S$ is in the vector representation, i.e., it includes x-y coordinates of road lanes points. In the case of a rasterized scene, an RGB value is provided for each pixel of the image. Therefore, it is the same as the vector representation unless here we have information (RGB value) about other parts of the scene in addition to the lanes. Hence, the transformation function can be applied directly on all pixels of the image. In other words, in image representation, $s$ is the coordinate of each pixel which has an RGB value and $\hat{s}$ represents the new coordinate with the same RGB value as $s$. 
Figure 9. **Retrieving real-world places using our real-world retrieval algorithm.** We observe that the model fails in Paris (a), New York (b), Hong Kong (c) and New Mexico (d). The model also successfully predicts in the drivable area in the remaining figures.

### D. Hyperparameters

In the paper, we used the following parameters $\alpha_1 = \beta_{11} = 10$, $\alpha_3 = \beta_{13} = 3$, $\beta_2 = 10$ and $\gamma_2 = 1/60$ as fixed values. The remaining parameters are defined as the transformation power $\alpha_2 \times 3000$, $\beta_{12} \times 3000$ and $\gamma_1$ for smooth, double and ripple scene generations, respectively. The test range for the transformation power is all integer values between $-9$ and $9$.

The hyperparameters for the new black-box algorithms
Figure 10. The predictions of different models in some generated scenes. All models are challenged by the generated scenes and failed in predicting in the drivable area.

Figure 11. Some examples showing the noise in the drivable area map. All these predictions were considered as off-road because of an inaccurate drivable area map.
Figure 12. The animations show the changes of the models predictions in different scenes. It is best viewed using Adobe Acrobat Reader.

Table 5. Comparing the performance of different baselines in the original dataset scenes and our generated scenes after removing trivial scenarios. SOR and HOR are reported in percent and the lower represent a better reasoning on the scenes by the model. Numbers are rounded to the nearest integer.

| Model       | Original SOR / HOR | Smooth-turn SOR / HOR | Double-turn SOR / HOR | Ripple-road SOR / HOR | All SOR / HOR |
|-------------|-------------------|-----------------------|-----------------------|-----------------------|---------------|
| DATF [38]   | 1 / 2             | 44 / 92               | 43 / 91               | 50 / 95               | 51 / 99       |
| WIMP [27]   | 0 / 1             | 30 / 80               | 23 / 71               | 29 / 77               | 31 / 82       |
| LaneGCN [31]| 0 / 1             | 23 / 65               | 32 / 75               | 34 / 77               | 37 / 81       |
| MPC [55]    | 0 / 0             | 0 / 0                 | 0 / 0                 | 0 / 0                 | 0 / 0         |

Table 6. Comparing the performance and computation time of different optimization algorithms in the generated scenes.

| Optimization algorithm | on LaneGCN [31] SOR / HOR | GPU Hours |
|------------------------|---------------------------|-----------|
| Baysian [41, 47]       | 13 / 40                   | 17.5      |
| GA [34]                | 14 / 45                   | 25.0      |
| TPE [9]                | 14 / 45                   | 12.1      |
| Brute force            | 23 / 66                   | 4.2       |

are considered as follows. In Baysian optimization [41, 47] and Tree-structured Parzen Estimator Approach (TPE) [9], the iteration number is set to 40 (after evaluating 10 random points). In Genetic algorithm [34], the population size is equal to 20, number of generations is set to 20, and mutation probability is 0.25.