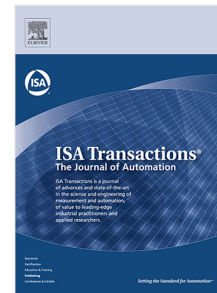


## Journal Pre-proof

Towards robust autonomous driving systems through adversarial test set generation

Devrim Unal, Ferhat Ozgur Catak, Mohammad Talal Houkan,  
Mohammed Mudassir, Mohammad Hammoudeh



PII: S0019-0578(22)00599-7  
DOI: <https://doi.org/10.1016/j.isatra.2022.11.007>  
Reference: ISATRA 4859

To appear in: *ISA Transactions*

Received date: 5 April 2022  
Revised date: 9 November 2022  
Accepted date: 9 November 2022

Please cite this article as: D. Unal, F.O. Catak, M.T. Houkan et al., Towards robust autonomous driving systems through adversarial test set generation. *ISA Transactions* (2022), doi: <https://doi.org/10.1016/j.isatra.2022.11.007>.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2022 ISA. Published by Elsevier Ltd. All rights reserved.

Revised Manuscript (Clean version)

1 Highlights

2 **Towards robust autonomous driving systems through adversarial test set generation**

- 3 • A new method for uncertainty-based test set generation for autonomous systems  
4 • Support training more robust DL models for autonomous driving  
5 • Improved model performance in the existence of data perturbations and adversarial attacks

Journal Pre-proof

# 6 Towards robust autonomous driving systems through adversarial 7 test set generation

8

## 10 ARTICLE INFO

11 *Keywords:*  
12 risk-aware autonomous systems  
13 DL  
14 test set generation  
15 uncertainty  
16

## ABSTRACT

Correct environmental perception of objects on the road is vital for the safety of autonomous driving. Making appropriate decisions by the autonomous driving algorithm could be hindered by data perturbations and more recently, by adversarial attacks. We propose an adversarial test input generation approach based on uncertainty to make the machine learning (ML) model more robust against data perturbations and adversarial attacks. Adversarial attacks and uncertain inputs can affect the ML model's performance, which can have severe consequences such as the misclassification of objects on the road by autonomous vehicles, leading to incorrect decision-making. We show that we can obtain more robust ML models for autonomous driving by making a dataset that includes highly-uncertain adversarial test inputs during the re-training phase. We demonstrate an improvement in the accuracy of the robust model by more than 12%, with a notable drop in the uncertainty of the decisions returned by the model. We believe our approach will assist in further developing risk-aware autonomous systems.

24

## 25 1. Introduction

26 Over the past decade, Deep Learning (DL) succeeded in a wide range of applications, partly due to the use of powerful learning algorithms that can learn complex relationships from large-scale datasets. More recently, DL schemes were applied successfully in various tasks related to autonomous driving, such as perception, prediction, planning, decision-making, and control. Developing robust DL schemes continues to attract more attention to safety-critical application domains. Despite the success of DL[1], there are still challenges in applying DL to many real-world applications. One of the biggest obstacles to using DL in many applications is the lack of explainability of how and why DL networks work.

33 In some cases, e.g., in medical applications, the neural network function must be explained to the user [2]. Another critical issue is that Deep Neural Networks (DNNs) are very sensitive to the change in the distribution of inputs. For example, the image classification network trained on the ImageNet dataset [3] can not be directly used to classify images of different domains, such as medical images. If we want to use a model trained on a large dataset, e.g., ImageNet [4], to classify images in other domains, the model will not work well. The main challenge here, is to train a model that performs on all different data types and in real-life circumstances with high uncertainty and noise.

39 When deploying a model to a new environment, it is critical to test the model to ensure accurate results. However, there are many ways to test a model. Traditional testing methods, including statistical testing, modeling, and validation, do not produce a robust enough test set that can cover all possible scenarios. Instead, to build DL models ready for industrial applications, we need to follow software test engineering methodologies [5] to develop and validate robust ML models. There are many different testing methodologies in software testing, such as white-box testing, black-box testing, and grey-box testing [6]. The grey-box testing is a methodology that combines white-box testing and black-box testing. Grey-box testing involves both the internal structure of the system and the environment in which the system operates. The system's internal structure is used to generate test data, and the environment is used to validate the test. Grey-box testing is a robust testing methodology that can test the system from different perspectives.

48 In this paper, we propose a grey-box testing methodology that can test the system from the internal structure of the system, the environment, and the user. The proposed method uses a system model to generate test data, a test oracle to validate the test, i.e., prediction performance, and a user model to evaluate the system, i.e., DNN model. There are different approaches to developing robust DL schemes, such as modifying the data, the model itself, adding auxiliary models, and adversarial test input generation. In this work, our focus is the test input generation-based robust DL scheme development in autonomous driving systems. Two significant gaps in recent works are the interpretability of DL models and training robust models in the presence of different training and test distributions [7]. This paper

ORCID(s):

55 addresses both of these gaps by introducing a well-defined adversarial test set generation that reduces the uncertainty  
56 and produces more robust models that perform better in the presence of uncertain inputs.

57 We consider the two main types of uncertainty: epistemic (model) and aleatoric (data) uncertainty. When au-  
58 tonomous driving is considered, DL models are generally multi-output regression models. Classical means of dealing  
59 with uncertainty generally do not apply to DL. There is a need for novel methods for measuring uncertainty in multi-  
60 output regression models. According to Wickramasinghe's recent work [8], many DNN solutions for Cyber-Physical  
61 System (CPS) data only create outputs for the input instances. They do not measure uncertainty in the prediction phase,  
62 which could ultimately result in a lack of trust in these approaches and could lead to unexpected and dangerous CPS  
63 behaviors. DNN models trained on data from CPS will require an uncertainty quantification framework to quantify  
64 the model uncertainty correctly. Next, the researchers should develop procedures to deal with this uncertainty.

65 One of the open problems for training robust models against uncertainty is generating relevant testing data to  
66 optimize the models. For this purpose, we use existing autonomous driving data sets, including sensor data, e.g.,  
67 LIDAR and GPS, and video recording data. We generate highly uncertain test inputs from these data sets to train  
68 robust models against perturbations and adversarial attacks. The generated test data will help reduce the uncertainty  
69 of the decision-making process of an autonomous driving system. To evaluate the robustness of the produced models,  
70 we use the generated test data to test the trained models. We also compare the results with the models which are  
71 not trained with the generated test data. Our recent work [9] presented a new metric for uncertainty quantification for  
72 object detection DNN models, where the experiments were conducted on the NEXET, Berkeley DeepDrive, KITTI, and  
73 Stanford datasets, together with SSD300, SSD512, and YoLo DNN models, to quantify prediction-time uncertainty.  
74 Another study [10] proposed a robust model training method (NIRVANA). NIRVANA seeks to compare the accuracy  
75 of DNN model predictions with another model and to enhance DNN model predictions using results of generated  
76 uncertainty quantification.

77 Connected and Autonomous Vehicles (CAVs) are subjected to cyber-attacks, particularly, zero-day attacks [11].  
78 The limited computational power of Engine Control Units (ECU) in cars prohibits them from processing robust secu-  
79 rity protocols. Furthermore, cyber-attacks put the privacy and security of passenger data at risk. CAVs differ from  
80 traditional software in several ways including their cyber-physical nature, which comprises hardware, software, and  
81 physics. The complex operating environment of CAVs includes people, and communication platforms [12]. The de-  
82 velopment and operation of CAVs are also fraught with uncertainty due to the inherent uncertainty of the DL models,  
83 the naturally unpredictable nature of the environment, the unpredictable nature of human behavior, and the unreliable  
84 network communications among CAV parts [13, 14]. The building and use of CAVs are challenging because of such  
85 uncertainty sources. Therefore, the data generated during the operation of CAVs can be used with DL models to under-  
86 stand CAV behaviors' uncertainties better, improve future generations of CAVs, and build novel test cases. We focus  
87 on the CAV data generated during their operation to achieve this.

88 There are many studies in the literature on detecting and preventing poisoning and trojan attacks; however, these  
89 studies mainly require using powerful computing resources and the cloud. Our study is different from others because we  
90 focus on test data set generation for robust ML/AI models for autonomous driving, including the development of low-  
91 energy, low-latency, and low-complexity methods that can easily integrate into the training and inference algorithms  
92 without affecting their performance.

93 The main contributions of this article are as follows:

- 94 1. We propose a method for generating highly uncertain test input data for a given DL model for autonomous  
95 driving systems. The generated test data are used to verify the robustness of DL models.
- 96 2. We adopted an adversarial training mitigation method for the highly uncertain test inputs to increase the robust-  
97 ness of the DL models.
- 98 3. We compare the results of our proposed method with the state-of-the-art testing method and show that our  
99 approach achieves better performance.

100 The rest of the paper is organized as follows. In Section 2, we present the related work. Section 3 presents the  
101 system model including the uncertainty-based test set generation and the adversarial re-training algorithm. Section 4  
102 presents the experiments on generating test inputs based on uncertainty maximization. In Section 5, we present the  
103 results and discuss the outcomes. We conclude with Section 6, where we also give ideas on future research tracks.

## 104 2. Related Work

105 DL recently gained popularity in autonomous driving and intelligent transportation system-related problems, with  
106 the ability to conduct feature selection during the learning process [15]. Some popular DL algorithms include Convo-  
107 lutional Neural Network (CNN), Feedforward Deep Network (FDN), Long-Short Term Memory (LSTM), Recurrent  
108 Neural Network (RNN), and Generative Adversarial Network (GAN) [15]. Boukerche et al. [16] discuss the appli-  
109 cation of DL to vision-based autonomous vehicle recognition. Various use cases include vehicle detection, vehicle  
110 make and model recognition, and vehicle re-identification. Khan et al. present [17] an LSTM-based intrusion detec-  
111 tion system for the Internet of vehicles. Jebamikyous and Kashef [18] discuss the two prominent use cases of DL for  
112 perception in autonomous vehicles; semantic segmentation and object detection. These tasks require precise data for  
113 high-performance classification results. However, uncertainty is a factor that reduces the classification performance  
114 of DL algorithms.

115 Various research works addressed the uncertainty in autonomous systems and proposed multiple methods to in-  
116 crease the performance of ML/DL algorithms in the presence of uncertainty. In [19], the authors analyze various  
117 sources of uncertainty in autonomous systems, such as execution uncertainty, timing uncertainty, adversarial attacks,  
118 and uncertainty of the behavior of Neural Networks (NNs). The authors propose a cross-layer weakly-hard design  
119 framework to tackle the execution uncertainty problem and safety verification of NNs through reachability analysis.  
120 Another study [20] addresses safe and adaptive autonomous navigation in uncertainty. The common shortcoming of  
121 the approaches mentioned above is that they require knowledge of the NN architecture to improve performance. In our  
122 proposed work, test-set generation based on uncertain inputs and subsequent re-training is utilized for this purpose,  
123 and previous knowledge of the NN architecture is not needed.

124 Uncertainty quantification is a method to rank the generated inputs based on their corresponding uncertainty metrics  
125 values (variance, maximum probability), re-training the model with the highly uncertain new generated inputs. In [21],  
126 the authors apply the Monte-Carlo (MC) dropout method to measure uncertainty in inputs for autonomous driving.  
127 The deep ensemble is a non-Bayesian uncertainty quantification method [22]. Adversarial ML methods are based on  
128 generating challenging and misleading inputs using attacks and then adding generated data or images to the original  
129 test set. In [23], Ma et al. use various adversarial training methods, such as the DeepFool, Basic Iterative Method,  
130 Carlini-Wagner (CW), Fast Gradient Sign Method (FGSM), and the Jacobian-Based Saliency Map Attack. The main  
131 aim of this study is to increase the uncertainty in the re-training phase to improve the model prediction performance.

132 In DeepXplore, new metrics such as Network Coverage (NC) and the number of activated neurons in the predic-  
133 tion have been used [24]. DeepTest generates new inputs for autonomous driving DL models where different image  
134 transformations lead to different NC values [25]. DeepMutation uses mutation score, a parameter used in traditional  
135 software engineering for testing DNN models. Their proposed method creates multiple new mutant DNN models from  
136 a given model using additional methods like neuron switch or layer removal [26]. DeepCT is a test coverage metric  
137 indicating that within a given DNN layer, all tuples of neurons in that DNN layer must be covered by at least one test  
138 image [27]. DeepHunter is a fuzzing-based test generation algorithm to hunt defects in DL models. The coverage  
139 metrics guide the fuzzing [28]. Ma et al. [29] define uncertainty-based metrics for test case generation in DL systems.  
140 The authors argue that the most critical test inputs are those with high uncertainty, a principle we adopt in this paper  
141 for generating more realistic autonomous driving test sets.

142 Djenouri et al. proposed an accurate object detection model by adopting the Granular Region Convolution Neural  
143 Network (GRCNN) to process vehicle image data to reduce accidents through a smart road system [30]. Mekala et  
144 al. highlighted the importance and feasibility of DL techniques for autonomous vehicles to accomplish an intelligent  
145 driving system without human interaction [31]. They have reviewed the Lidar-based DL strategies to address the  
146 research challenges in autonomous driving through a comprehensive analysis of Semantic Segmentation, Data Rep-  
147 resentation, Feature Extraction, Dynamic Object Detection, Data Fusion, and Autonomous Driving-Multi-Objective  
148 tracking mechanisms.

149 The existing literature addresses uncertainty quantification and various methods to train better-performing ML/AI  
150 models in the presence of uncertainty. The current literature lacks the issue of training robust models against uncer-  
151 tainty by generating relevant testing data to optimize the models. There is no systematic approach to test data generation  
152 based on adjustable parameters that can be measured which represents the uncertainty of the test data, and evaluates  
153 the effect of changing these parameters on the performance.

Towards robust autonomous driving systems through adversarial test set generation

	Bayesian Methods	Non-Bayesian Methods
Probabilistic	- Variational Inference (VI) - Markov Chain Monte Carlo (MCMC)	- Monte Carlo Dropout (MCD) - Bootstrap Ensemble
Non-Probabilistic	- None	- Confidence Interval (CI) - Importance Weighting (IW)

Table 1

An overview of methods for measuring output uncertainty.

### 3. Uncertainty Types and Quantification in DL

To mitigate the overconfidence problem in DL models, we need to train the model to be uncertain about its predictions. This section introduces methods from the literature used to measure the output uncertainty of DL models. Table 1 shows the two main types of methods for measuring the output uncertainty of DL models. Bayesian methods are based on Bayesian inference, whereas non-Bayesian methods are based on other methods such as Monte Carlo sampling, importance sampling, and bootstrap sampling. Bayesian methods are usually more accurate but require more extensive training data and are computationally expensive compared to non-Bayesian methods. For these reasons, non-Bayesian methods are more prevalent in practice.

We present our view of the taxonomy of uncertainty types in Fig. 1. This taxonomy is motivated by the notion of *epistemic* and *aleatoric* uncertainty in the statistical literature and the recent work on DL uncertainty in [32]. The boxes in the diagram show various sources of uncertainty, while the arrows indicate how they propagate through the DL model. The middle region shows the sources of uncertainty considered in this paper.

*Epistemic* uncertainty, also known as model uncertainty, arises from the fact that the model has been trained on an imperfect dataset and in an imperfect environment. It corresponds to the model's inherent uncertainty, which one can reduce during training by providing more data. *Aleatoric* uncertainty, or data uncertainty, stems from imperfect and incomplete data. Whereas *Classical* uncertainty, or label uncertainty, arises from the fact that the ground truth labels are imperfect and incomplete. Classical uncertainty corresponds to the inherent uncertainty in the labels, which can be reduced by providing more data. *Prediction* uncertainty, or output uncertainty, arises from the fact that the model has been trained on an imperfect dataset and in an imperfect environment. It corresponds to the inherent uncertainty in the model, which can be reduced during training by providing more data.

Model, prediction, and data uncertainty can all be exploited to improve the quality of predictions made by the model. Epistemic and Prediction uncertainty are the *reducible* components of the total uncertainty. While Classical and Aleatoric uncertainty can be considered as the *irreducible* component of the total uncertainty.

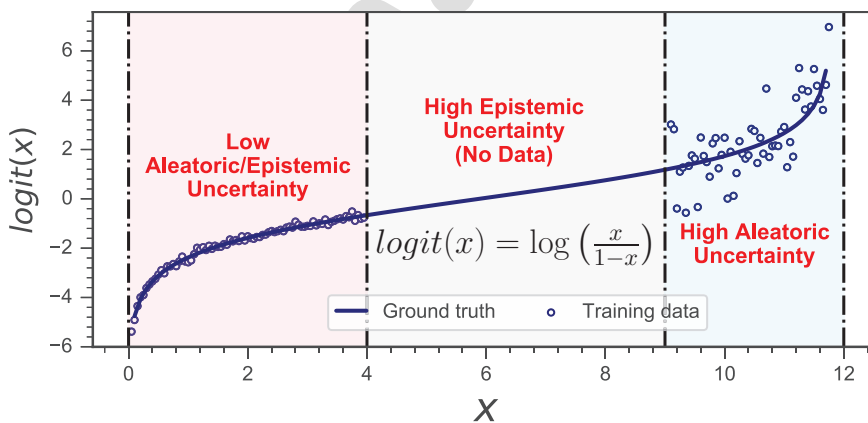


Figure 1: Illustration of the different sources of uncertainty and how they affect the performance of a DL model.

DL models are often trained using a *maximum likelihood* ML criterion, i.e., the model parameters are learned to

Towards robust autonomous driving systems through adversarial test set generation

maximize the likelihood of the training data. However, the ML criterion is known to be the Maximum a Posteriori (MAP) estimate in the Bayesian setting [33]. Thus, the ML criterion can be viewed as a special case of the MAP criterion when the prior is uniform and can be used to learn the model parameters of a DL model. However, the MAP criterion does not account for the uncertainty in the model parameters. Instead, it is accounted for by using the MAP criterion. The MAP criterion is given by

$$\theta^* = \arg \max_{\theta} \log p(\mathcal{D} | \theta) + \log p(\theta) \quad (1)$$

177 where  $\mathcal{D}$  is the training data,  $\theta$  are the model parameters,  $p(\mathcal{D} | \theta)$  is the likelihood of the training data given the model  
178 parameters, and  $p(\theta)$  is the prior over the model parameters.

The uncertainty in the training data can be accounted for by using the *maximum a posteriori with data uncertainty* (MAP-DU) criterion defined as

$$\theta^* = \arg \max_{\theta} \log p(\mathcal{D} | \theta) + \log p(\theta | \mathcal{D}) \quad (2)$$

179 where  $\mathcal{D}$  is the training data,  $\theta$  are the model parameters,  $p(\mathcal{D} | \theta)$  is the likelihood of the training data given the model  
180 parameters, and  $p(\theta | \mathcal{D})$  is the posterior over the model parameters given the training data.

The MAP-DU criterion can be used to learn the model parameters of a DL model. However, it does not account for the uncertainty in the labels. The label uncertainty can be accounted for by using the *maximum a posteriori with label uncertainty* (MAP-LU) criterion defined as

$$\theta^* = \arg \max_{\theta} \log p(\mathcal{D} | \theta, \mathcal{L}) + \log p(\theta | \mathcal{D}, \mathcal{L}) \quad (3)$$

181 where  $\mathcal{D}$  is the training data,  $\mathcal{L}$  are the labels,  $\theta$  are the model parameters,  $p(\mathcal{D} | \theta, \mathcal{L})$  is the likelihood of the training  
182 data given the model parameters and labels, and  $p(\theta | \mathcal{D}, \mathcal{L})$  is the posterior over the model parameters given the  
183 training data and labels.

The MAP-LU criterion does not account for the uncertainty in the test data. Hence, the uncertainty in the test data can be accounted for by using the *maximum a posteriori with the test data uncertainty* (MAP-TDU) criterion given by

$$\theta^* = \arg \max_{\theta} \log p(\mathcal{D}_{\text{test}} | \theta, \mathcal{D}) + \log p(\theta | \mathcal{D}, \mathcal{D}_{\text{test}}) \quad (4)$$

184 where  $\mathcal{D}$  is the training data,  $\mathcal{D}_{\text{test}}$  is the test data,  $\theta$  are the model parameters,  $p(\mathcal{D}_{\text{test}} | \theta, \mathcal{D})$  is the likelihood of the  
185 test data given the model parameters and training data, and  $p(\theta | \mathcal{D}, \mathcal{D}_{\text{test}})$  is the posterior over the model parameters  
186 given the training data and test data.

187 DL models are trained to optimize a loss function, making them susceptible to the data used to train the model.  
188 The dependence on the training data may significantly impact the model's generalization performance on unseen data  
189 during testing because DL models can extrapolate the underlying relationships in the data and generalize them to  
190 unseen data. However, a model trained on a dataset sampled from a different distribution from the testing one will not  
191 generalize well.

192 Figure 1 shows the three sources of uncertainty and how they affect the performance of a DL model. Data un-  
193 certainty comes from the fact that the dataset is sampled from an underlying distribution. Irreducible uncertainty is  
194 uncertainty that one can not remove. In contrast, model uncertainty is the uncertainty that comes from the fact that the  
195 model is not trained on the entire population. Figure 1 shows how the three sources of uncertainty propagate through  
196 the model. The data uncertainty affects the model uncertainty, which affects the irreducible uncertainty. A model  
197 trained on a dataset generalizes better if the dataset is larger. This is because a model trained on a larger dataset is more  
198 accurate, which reduces the model's uncertainty. Data uncertainty comes from the fact that the dataset is sampled from  
199 an underlying distribution. The data uncertainty affects the model uncertainty, which in turn affects the irreducible  
200 uncertainty. The model uncertainty is the primary source of uncertainty that we focus on in this work.

201 A model with a high uncertainty has low performance on unseen data. In this work, we focus on the aleatoric  
202 uncertainty. There are two main ways to quantify aleatoric uncertainty, the mean-variance approach, and the predictive  
203 entropy approach. The mean-variance method quantifies the aleatoric uncertainty by the model's predictions' mean  
204 and variance. The predictive entropy approach quantifies the aleatoric uncertainty by the entropy of the model's  
205 predictions. In this article, we use the mean-variance approach.



Towards robust autonomous driving systems through adversarial test set generation

206 In the mean-variance approach, the mean is the expected value of the model's predictions, and the variance is the  
 207 expected value of the squared difference between the model's predictions and the mean. The mean and the variance  
 208 can be estimated using the Monte Carlo method [34]. The Monte Carlo method generates samples from the model's  
 209 predictions. The mean and the variance are then estimated using the generated samples.

## 210 4. System Model

211 The main goal of this work is to build a robust DL model for Cyber-Physical Systems (CPSs), based on model  
 212 re-training creating highly uncertain inputs. The proposed highly uncertain instance generation mainly consists of the  
 213 following three phases: (I) is the dataset generation; (II) is the building of a dropout-based neural network at prediction  
 214 time; (III) is the re-training of the model with the generated highly uncertain inputs (see Figure 2).

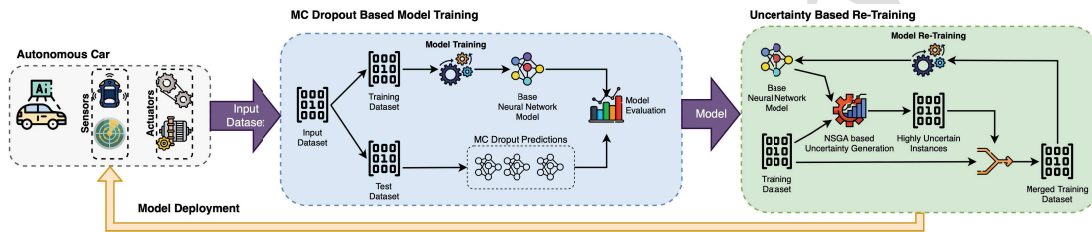


Figure 2: System overview.

### 215 4.1. Test Input Generation based on the Non-dominated Sorting Genetic Algorithm

216 NSGA-II (Non-dominated Sorting Genetic Algorithm-II) follows the general outline of a genetic algorithm with a  
 217 modified mating and survival selection. In this algorithm individuals are chosen frontwise, leading to a situation where  
 218 a front needs to be split because not all individuals are permitted to survive. This splitting front selects solutions based  
 219 on crowding distance [35]. The main advantages of NSGA-II are that it can deal with mixed constraints, it is elitist, is  
 220 very unlikely to converge prematurely to a local optimum, and is computationally efficient. Elitist algorithms, a type of  
 221 evolutionary algorithm, work by making sure that the best individuals in one epoch are not discarded, and transferred  
 222 directly into the next generation. The algorithm is implemented in Python through the Pymoo module. Algorithm 1  
 223 shows the steps in NSGA-II. The algorithm's inputs are; the population size  $P$ , the number of generations  $G$ , the  
 224 objective function  $f_i(x_i)$ , the lower and upper bounds  $l_i, u_i$  of decision variables  $x_i$ . The output of NSGA-II is the set  
 225 of non-dominated solutions  $S^*$ .

---

#### Algorithm 1 NSGA-II

---

**Require:**  $P, G, f_i(x_i), l_i, u_i$

**Ensure:**  $S^*$

- 1: Initialize the population  $P$ ;
  - 2: **for**  $i = 1$  to  $G$  **do**
  - 3:   Calculate the fitness of each individual  $x_i$ ;
  - 4:   Sort the population  $P$  frontwise;
  - 5:   Select the non-dominated individuals  $S^*$ ;
  - 6:   Calculate the crowding distance of  $S^*$ ;
  - 7:   Select the best individuals with the smallest crowding distance;
  - 8:   Generate the new population  $P_{new}$ ;
  - 9:    $P := P_{new}$
  - 10: **end for** RETURN  $S^*$
- 

226 Figure 2 demonstrates the training dataset  $D$  is collected from CAVs in Phase I. In this study the inputs are LIDAR,  
 227 and the output is a robust model that is trained to classify the objects in the LIDAR. The algorithm's primary output is



228 the generation of a robust model that is not tricked by an adversarial test input injection. Data preparation is conducted  
 229 in Phase II and DNN model training with the input dataset  $D$  which is carried out from the previous phase. Conventional  
 230 DNN training is executed at this phase to find the optimal weights. The primary output is the base DNN model, which  
 231 is the DNN model with the best prediction performance and lowest loss value over the complete training data. The  
 232 most crucial contrast between the base DNN model and a conventional DNN is the prediction time activated dropouts.  
 233 In contrast, a standard DNN model has no dropouts in the prediction time.

234 In Phase III, the model is re-trained using highly uncertain training instances to improve the DNN model's pre-  
 235 diction performance. In this our approach, we use the training dataset and model with the NSGA-II optimization  
 236 algorithm to generate new and highly uncertain instances. Then, we merge the training dataset with newly generated  
 237 instances. Finally, we re-train the DNN model with the new training dataset to improve the prediction performance.

## 238 4.2. Adversarial Training

239 In this research, we use the adversarial training method to improve the robustness of the DL model. The adversarial  
 240 training mitigation method re-trains a DNN model to be robust against a specific type of perturbation. In this work,  
 241 we train a model on a training set, where the inputs are perturbed by adding a small amount of noise generated by  
 242 the NSGA-II optimization-based perturbation method to increase the uncertainty of the test input for the DL model.  
 243 Algorithm 2 shows the proposed method for adversarial training. The algorithm's input is the dataset  $D$ , the number  
 244 of iterations  $I$ , the learning rate  $\alpha$ , the model  $F$ , and the loss function  $L$ . The output of the algorithm is the robust  
 245 model  $F'$ .

---

### Algorithm 2 Adversarial Training

---

**Require:**  $D, I, \alpha, F, L$

**Ensure:**  $F'$

- 1: Initialize the parameters of  $F$ ;
  - 2: Initialize the parameters of  $F'$ ;
  - 3: **for**  $i = 1$  to  $I$  **do**
  - 4: Randomly select an input  $x_i$  from  $D$ ;
  - 5:  $x_{unc} := x_i + \alpha \cdot NSGA - II(x_i, y_i, F)$ ;
  - 6: Update the model  $F$  by minimizing  $L(x_{unc}, y_i, F)$ ;
  - 7: Update the model  $F'$  by minimizing  $L(x_{adv}, y_i, F')$ ;
  - 8: **end for**
  - 9: **return**  $F'$
- 

## 246 5. Experimental Evaluation

247 The aim of our experiments is to assess the effectiveness of our approach in minimizing the effect of uncertainty  
 248 in the prediction results. The experimental setup and conditions are chosen to ensure that the results are as close  
 249 to real-world conditions as possible. The results are then used to improve the model, and the process is repeated  
 250 until the desired level of accuracy is achieved. We determined the best hyper-parameters for the DNN models for the  
 251 datasets used in the experiments using a simple grid search. We found that the best hyper-parameters are the **ADAM**  
 252 optimization with a learning rate of 0.001. Figure 3 shows the DNN model. Dropout layers have been used during the  
 253 model's training against adversarial test input injection. ReLu activation is used in the input and all the hidden layers;  
 254 the output layer is activated by the Sigmoid function with the loss given by categorical cross-entropy. We perform the  
 255 optimization using the Adam optimizer. The data is available from <sup>1</sup>. We present information about the dataset we  
 256 employed in the experiments in Section 5.2, followed by the training process description.

### 257 5.1. Generation of Challenging Inputs Based on Uncertainty Maximization

258 Our approach is based on generating challenging inputs based on uncertainty maximization. Our method is similar  
 259 to the methods utilized for defending against adversarial ML attacks. Here, we try to increase the prediction uncertainty  
 260 of the model by adding a small perturbation to the input. In adversarial ML attacks, the objective is to fool the model  
 261 using the loss maximization approach. We extend this approach by maximizing the prediction uncertainty in the

<sup>1</sup><https://level-5.global/data/perception/>

Towards robust autonomous driving systems through adversarial test set generation

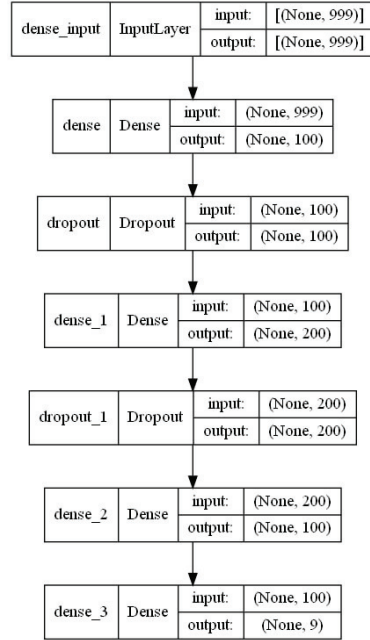


Figure 3: Architecture of the deep neural network.

262 model's prediction. We depict the overview of our methodology in Fig. 4. In the first step, the data is pre-processed  
 263 including imbalance correction and train-test split. The second step trains the deep neural network model is trained  
 264 on the train set. Once the model has sufficient performance, an adversarial test input generation is used to modify the  
 265 training dataset. In the third step, the model is re-trained on the adversarial dataset. The model becomes more robust  
 266 to this type of adversarial attack based on our approach.

267 To generate challenging inputs, i.e., highly uncertain inputs, we used an optimization method to find the best  
 268 perturbation points on the input to increase the uncertainty. The objective function of the optimization method is  
 269 to maximize the prediction uncertainty. There are three parts to the objective function, increase uncertainty, lower  
 270 perturbation amount, and increase wrong prediction amount. The objective function combines the hyper-parameters  
 271 ( $\alpha$  and  $\beta$ ) to find the optimal solution. The optimization objective is to create new instances based on training data  
 272 with the following optimization function given in Eq. 5. Eq. 5 is constrained by the noise budget  $\epsilon$ :  $\|\mathbf{noise}\| < \epsilon$ . We  
 273 have assigned  $\epsilon$  as 30% of the training dataset.

$$\mathbf{noise} = \underset{\mathbf{noise}}{\arg \min} \left( \frac{\alpha}{\mathbb{U}(h(\mathbf{x}))} + \|\mathbf{noise}\| + \beta \cdot \mathbb{1}(y \neq \hat{y}) \right), \quad \|\mathbf{noise}\| < \epsilon \quad (5)$$

274 where:

- 275 -  $\mathbb{U}(h(\mathbf{x}))$  is the prediction uncertainty
- 276 -  $\mathbf{x}$  is the input instance
- 277 -  $h$  is the DL model
- 278 -  $\|\mathbf{noise}\|$  is the norm (i.e. magnitude) value of the noise
- 279 -  $\epsilon$  is the Noise budget (i.e. the maximum distance between  $\mathbf{x}$  and  $\mathbf{x} + \mathbf{noise}$ )
- 280 -  $\mathbf{x} \in \mathbb{R}^m$  and  $\mathbf{noise} \in \mathbb{R}^m$
- 281 -  $\mathbf{noise}$  has the same number of columns as the input instance  $\mathbf{x}$
- 282 -  $\alpha$  is the multiplication factor of uncertainty quantification metrics value
- 283 -  $\beta$  is the multiplication factor of the number of incorrect predictions

Towards robust autonomous driving systems through adversarial test set generation

- 284 The ideal **noise** would have the following properties:  
 285 - Maximize the prediction uncertainty:  $\frac{\epsilon}{U(x)}$   
 286 - Minimize the noise magnitude (norm value):  $\|\text{noise}\|$   
 287 - Change the prediction class:  $\mathbb{1}(y \neq \hat{y})$   
 288 We use the generated data to train the model to increase its robustness.

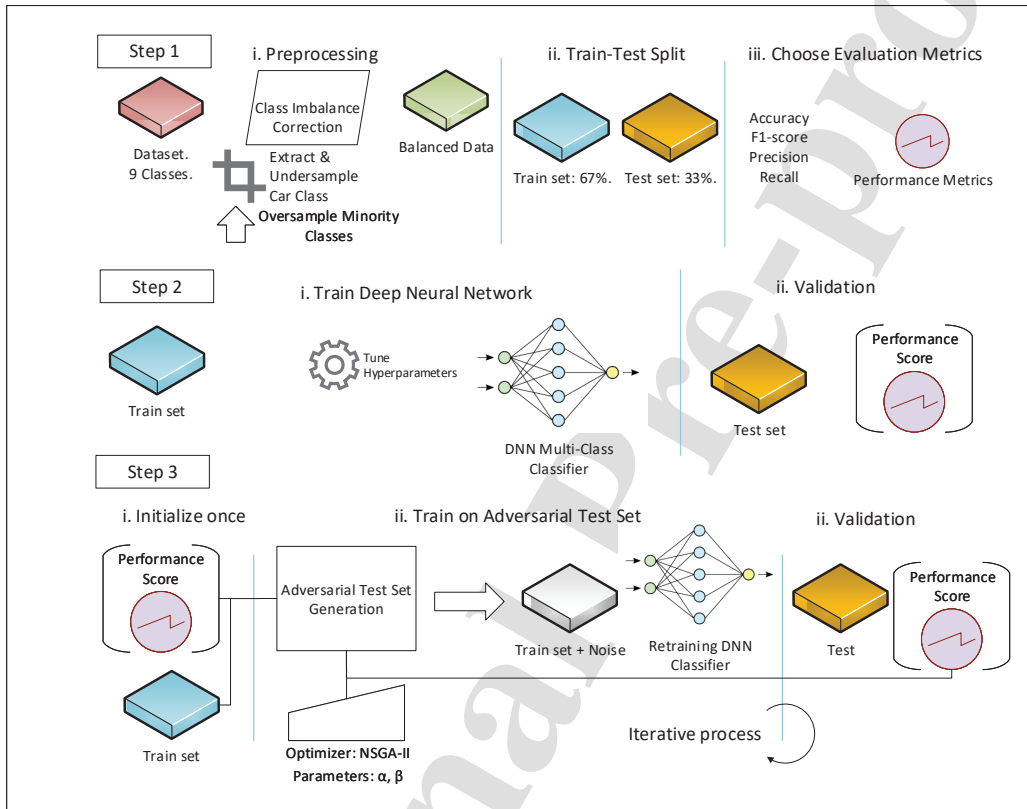


Figure 4: Steps of the uncertainty-based test input generation and DL model training.

## 289 5.2. Dataset

290 We use the LIDAR data in this study published by Woven Planet Holdings [36]. The data is available from  
 291 <https://level-5.global/data/perception>. The LIDAR dataset size is about 37GB. The dataset is collected as cloud points  
 292 as shown in Fig. 5. The cloud points are represented as 3D-coordinate points such as  $(x, y, z)$  where  $x$ ,  $y$ , and  $z$  are  
 293 floats. In this dataset, a LIDAR scene corresponds to 4 images. However, we do not consider images in this study. The  
 294 reason for using LIDAR data as opposed to images is that LIDAR sensors, unlike cameras, function independently of  
 295 ambient lighting. LIDAR can achieve great results both day and night without any loss of performance due to distur-  
 296 bances such as headlight glare, sunlight, shadows, or lack of ambient lighting. The LIDAR sensors use an eye-safe  
 297 laser to emit light pulses that light up the region of interest.

298 This dataset was initially released for regression models. However, we modified the dataset for training classifi-  
 299 cation models. Since the dataset is too large, we down-sampled to 10% of the original dataset. The LIDAR dataset  
 300 contains labels of the objects. We cropped the dataset and obtained the objects and their labels. The new dataset con-  
 301 sists of only LIDAR data of the objects and their labels. There are 9 types of objects: animals (anml), bicycle (bcycl),  
 302 bus, car, emergency vehicle (EV), motorcycle (MC), other vehicle (OV), pedestrian (pdstrn), and truck (trck). Only

Towards robust autonomous driving systems through adversarial test set generation

303 these objects are extracted to prepare the train and test sets. There is a class imbalance issue as there are more cars  
 304 than cyclists or emergency vehicles. Therefore, we also applied class imbalance correction. The minority classes are  
 305 oversampled through imbalance correction and the majority (car) class is undersampled. The training dataset is 67%  
 306 and the testing dataset is 33%. The ML training is initiated by training on LIDAR data and labels. The model outputs  
 307 the *softmax* classification of the objects. Then, using the uncertainty formula and NSGA-II algorithm, the dataset is  
 308 modified with noise and sent to the ML models again for re-training.

### 309 5.3. Training

310 For defining the models and visualizing the analysis using Python, the following packages are used: TensorFlow,  
 311 uncertainty wizard, Pandas, NumPy, Sci-Kit learn, SciPy, Keras, Seaborn, Pymoo, CleverHans, Tqdm, and Matplotlib.

312 The model is defined as a Stochastic Sequential Keras model with dense and dropout layers. ReLU activations  
 313 are used for input and hidden layers with softmax activation in the final output. The neurons in the dense layers  
 314 are 100, 200, and 100, respectively. The dropout layers use 9% dropout. Adam optimizer is used with the loss function  
 315 as categorical cross-entropy. The model is executed and trained with early callbacks enabled.

316 A different type of neural network, Pointnet, is also used for validating our approach. Pointnet is a neural network  
 317 designed to work directly with point clouds while considering the permutation in-variance. [37].

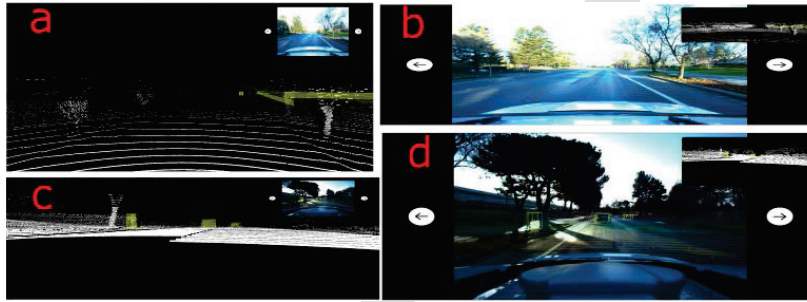


Figure 5: In this figure, (a) shows a LIDAR output, and its corresponding image in visible light is shown in (b). (c) is the LIDAR output of a different scene, and the corresponding visible light image is shown in (d).

## 318 6. Results and Discussions

319 The performance of the ML model is evaluated using accuracy (defined in Eq. 6), F-1 score (defined in Eq. 9), recall,  
 320 and precision. These metrics are determined using True Positives (TP), False Positives (FP), True Negatives (TN), and  
 321 False Negatives (FN). The F1-score is calculated based on precision and recall and a value of the F1-score close to 1  
 322 indicates that the model performs well in both precision (see Eq. 7) and recall (see Eq. 8).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

$$F1\text{-score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (9)$$

Towards robust autonomous driving systems through adversarial test set generation

323 Fig. 6 plots losses and accuracy as the model is trained against the adversarial test input injection dataset. We see  
 324 that accuracy is improving with a reduction in losses. Fig. 7 shows the accuracy, precision, recall, and F1 score during  
 325 the adversarial training. The performance metrics are improving with every training iteration. Fig. 8 shows the drop  
 in uncertainty before and after training with the adversarial inputs.

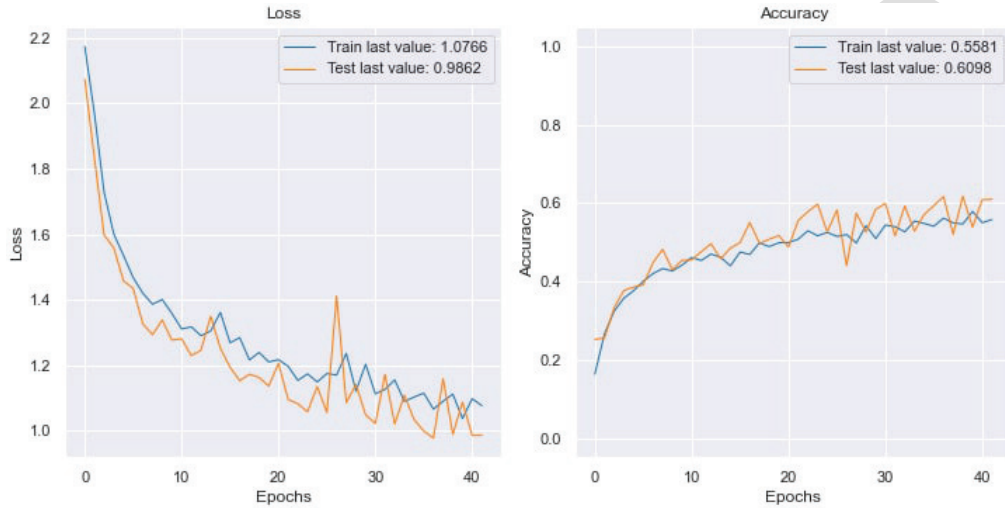


Figure 6: Loss plot training

326 The model is made more robust with adversarial ML-based test input generation. After we append the dataset with  
 327 the adversarial inputs by the allocated percentage of the noise budget, we re-train our model. In Fig. 7 we observe the  
 328 model performance drops initially but improves again and becomes more robust with subsequent training iterations.  
 329 Fig. 8 shows the drop in uncertainty after training with adversarial test inputs. Fig. 9 shows the improvement of the  
 330 model across all the performance metrics after training using the adversarial test input set. The Pointnet performs  
 331 better than the typical dense neural network as the architecture of this neural network is specially designed to directly  
 332 handle point clouds.  
 333

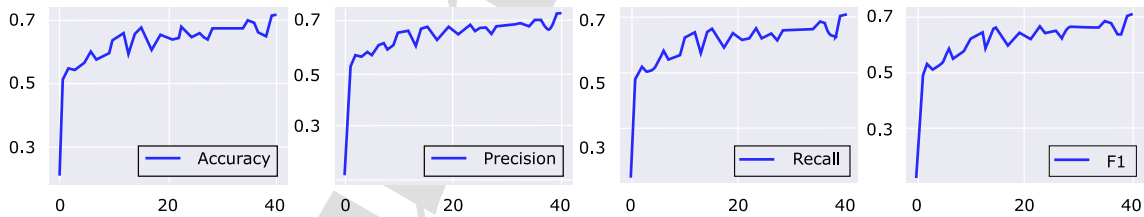


Figure 7: Accuracy, precision, and recall with adversarial test input injection training. Model performance improves with iterations.

334 Table 2 shows the influence of  $\alpha$  and  $\beta$  hyper-parameters on the performance of the DNN during the training  
 335 against adversarial test input injection.  $\alpha$  is the multiplication factor of uncertainty quantification metrics value. We  
 336 used mean variance as the uncertainty quantification metric.  $\beta$  is the multiplication factor of the number of incorrect  
 337 predictions. Varying these hyper-parameters will determine how much of the noise originates from the magnitude of  
 338 the prediction error, or the number of wrong predictions. The hyperparameter values depend on the type of data and  
 339 the classification task. Our optimization function uses these two hyper-parameters to penalize the classifier and train  
 340 the model to become robust to adversarial test input injection. The best value for  $\alpha$  and  $\beta$  is 0.6. The model showed

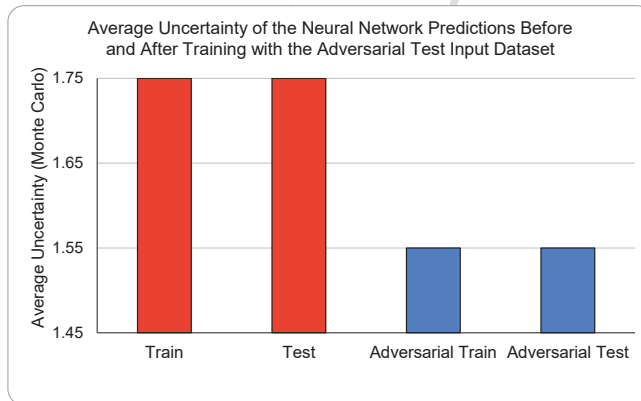
Towards robust autonomous driving systems through adversarial test set generation

**Table 2**

The table shows the values of the  $\alpha$  and  $\beta$  parameters and the resulting model performance, the % improvement compared to the base model, and the average performance metrics

$\alpha$	$\beta$	Accuracy	Precision	Recall	F1-score	$\alpha/\beta$	Acc. Improved
0.9	0.9	0.68	0.69	0.68	0.68	1	7.3
0.9	0.6	0.67	0.67	0.68	0.67	1.5	5.9
0.9	0.3	0.69	0.70	0.68	0.69	3	8
0.9	0.1	0.62	0.61	0.63	0.62	9	1.53
0.6	0.9	0.68	0.69	0.67	0.68	0.67	7.33
0.6	0.6	0.73	0.76	0.73	0.74	1	12
0.6	0.3	0.61	0.59	0.61	0.60	2	-0.27
0.6	0.1	0.65	0.66	0.64	0.65	6	4.06
0.3	0.9	0.60	0.62	0.61	0.61	0.33	-0.59
0.3	0.6	0.71	0.71	0.70	0.70	0.5	9.74
0.3	0.3	0.60	0.60	0.61	0.60	1	-0.64
0.3	0.1	0.69	0.70	0.68	0.69	3	8.16
0.1	0.9	0.72	0.71	0.70	0.70	0.11	11
0.1	0.6	0.63	0.62	0.61	0.61	0.17	2.01
0.1	0.3	0.52	0.52	0.51	0.51	0.33	-9.36
0.1	0.1	0.61	0.60	0.61	0.60	1	0.08

341 a relatively optimal performance using this value which yields an average accuracy of 0.73, i.e., a 12% improvement  
 342 upon the base model. The accuracy of the base model is 0.62.

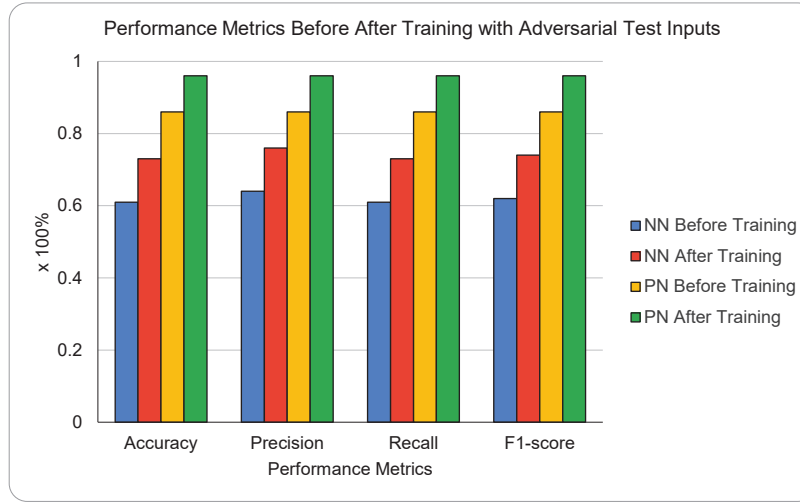


**Figure 8:** Average uncertainty of the Neural Network predictions before and after training on the dataset with the adversarial test inputs. Average uncertainty drops after training on the adversarial dataset.

343 Table 3 shows the class-wise prediction performance scores for the robust model. The accuracy, precision, recall,  
 344 and F1-score of the model show modest performance in absolute terms. The robust model exhibits relatively better  
 345 performance than the base model and is resistant to adversarial test input attacks.

346 In the context of autonomous driving, these results indicate that robust models would be better at correctly clas-  
 347 sifying the objects and will be secure against adversarial attacks with test input generation when compared against a

Towards robust autonomous driving systems through adversarial test set generation



**Figure 9:** The performance of the model improves across all metrics after training with adversarial test inputs. NN=dense neural network, PN=Pointnet

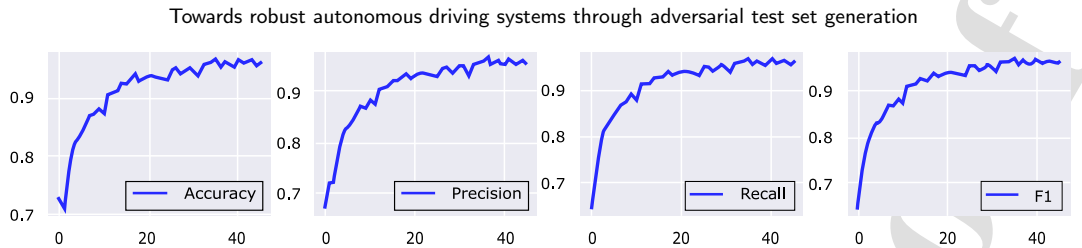
**Table 3**

The class-wise performance metrics of the DL model with the optimal  $\alpha$  and  $\beta$  trained against adversarial test input injection dataset

	animal	bcycl	bus	car	EV	MC	OV	pdstrn	trck
Accuracy	0.76	0.74	0.73	0.72	0.73	0.73	0.71	0.70	0.64
Precision	0.74	0.73	0.74	0.71	0.73	0.74	0.71	0.70	0.61
Recall	0.77	0.74	0.75	0.72	0.71	0.75	0.73	0.68	0.62
F1-score	0.75	0.73	0.74	0.71	0.72	0.74	0.72	0.69	0.61

348 base model that has not been trained likewise. The results also show that there is room for improvement across all  
 349 the metrics. Nevertheless, the robust model is already performing better than the base model by a margin of 12%.  
 350 Pointnet is used to classify point cloud data for the purpose of testing the adversarial test input injection training with  
 351 a better-suited model. The resulting model without the input injection the model had an accuracy of around 85%; after  
 352 the injection, the accuracy and other metrics reached around 96%. The improvement of 11% indicated that the method  
 353 could work on diverse model types. The performance of the trained robust Pointnet model across various metrics is  
 354 shown in Fig 10.





**Figure 10:** Accuracy, precision, and recall with adversarial test input injection training. the Pointnet Model, performance improves with iterations.

## 7. Conclusion and Future Works

Adversarial attacks are used maliciously to confuse ML models. Uncertainty, which is inherent in the training data, or arising from the training process, has a similar effect on the performance of ML models. This can have severe consequences in the case of autonomous vehicles where correct identification of objects in the surroundings and road is critical. In this work, we used an adversarial test input generation approach for making the ML model more robust against uncertainty and adversarial attacks. We demonstrated that adversarial test inputs can affect the performance of the ML model, which can have severe consequences, such as the mis-classification of objects on the road by autonomous vehicles. However, by building a dataset that contains adversarial test inputs and re-training, we can make the ML model more robust to adversarial attacks. The accuracy of the robust model can reach higher than that of the base model by up to 12%, with a scalar reduction of 0.25 in uncertainty.

Our main contribution is a new method based on the NGSA-II algorithm for generating highly uncertain test input data for a given DL model for autonomous driving systems. We increased the robustness of the DL models against high uncertainty using the adversarial training mitigation method. The presented approach performs better in producing robust DL models against similar works in the literature.

In future work, we plan to address data uncertainty problems in addition to the model uncertainty problem addressed in this article. For this purpose, we aim to investigate Kernel Density Estimation (KDE) based methods for moving the data instances in a training set into low-density regions, thus increasing the uncertainty of the training data. Another improvement to the current approach of generating uncertain test data is looking at a DL loss function-based method for generating data instances with high uncertainty. The NGSA-II based optimization used in this work exhibits good performance. However, NGSA-II takes a long time to generate highly uncertain examples. The DL loss function-based uncertainty maximization method has a similar performance with lower execution times. Therefore, we will investigate this method in our future works.

## References

- [1] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, A. A. Atayero, Smote-drrn: A deep learning algorithm for botnet detection in the internet-of-things networks, *Sensors* 21 (2021).
- [2] S. I. Popoola, B. Adebisi, M. Hammoudeh, H. Gacanin, G. Gui, Stacked recurrent neural network for botnet detection in smart homes, *Computers & Electrical Engineering* 92 (2021) 107039.
- [3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:doi:10.1109/CVPR.2009.5206848.
- [4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, *International journal of computer vision* 115 (2015) 211–252.
- [5] M. A. Jamil, M. Arif, N. S. A. Abubakar, A. Ahmad, Software testing techniques: A literature review, in: 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M), 2016, pp. 177–182. doi:doi:10.1109/ICT4M.2016.045.
- [6] Z. Hamza, M. Hammad, Testing approaches for web and mobile applications: An overview, *International Journal of Computing and Digital Systems* 9 (2020) 657–664.
- [7] A. Qayyum, M. Usama, J. Qadir, A. Al-Fuqaha, Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward, *IEEE Communications Surveys & Tutorials* 22 (2020) 998–1026.
- [8] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, M. Manic, Generalization of deep learning for cyber-physical system security: A survey, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 745–751. doi:doi:10.1109/IECON.2018.8591773.
- [9] F. Catak, T. Yue, S. Ali, Prediction surface uncertainty quantification in object detection models for autonomous driving, in: 2021 IEEE International Conference On Artificial Intelligence Testing (AITest), IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp. 93–100. URL: <https://doi.ieeecomputersociety.org/10.1109/AITEST52744.2021.00027>. doi:doi:10.1109/AITEST52744.2021.00027.

## Towards robust autonomous driving systems through adversarial test set generation

- 398 [10] F. O. Catak, T. Yue, S. Ali, Uncertainty-aware prediction validator in deep learning models for cyber-physical system data, *ACM Trans.*  
399 *Softw. Eng. Methodol.* (2022).
- 400 [11] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, O. Jogunola, Federated deep learning for zero-day botnet attack detection in  
401 *IoT-edge devices*, *IEEE Internet of Things Journal* 9 (2022) 3930–3944.
- 402 [12] E. Geisberger, M. Broy (Eds.), *Living in a Networked World: Integrated Research Agenda Cyber-Physical Systems (agendaCPS)*, acatech  
403 Studie, Utz, München, 2015. URL: [http://www.acatech.de/de/publikationen/empfehlungen/acatech/detail/artikel/  
404 living-in-a-networked-world-integrated-research-agenda-cyber-physical-systems-agendacps.html](http://www.acatech.de/de/publikationen/empfehlungen/acatech/detail/artikel/living-in-a-networked-world-integrated-research-agenda-cyber-physical-systems-agendacps.html).
- 405 [13] M. Zhang, S. Ali, T. Yue, R. Norgren, O. Okariz, Uncertainty-wise cyber-physical system test modeling, *Software & Systems Modeling* 18  
406 (2019) 1379–1418.
- 407 [14] T. Ma, S. Ali, T. Yue, Testing self-healing cyber-physical systems under uncertainty with reinforcement learning: an empirical study, *Empirical*  
408 *Software Engineering* 26 (2021) 1–54.
- 409 [15] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, M. Imran, Deep learning and big  
410 data technologies for IoT security, 2020. doi:doi:10.1016/j.comcom.2020.01.016.
- 411 [16] A. Boukerche, X. Ma, Vision-based autonomous vehicle recognition: A new challenge for deep learning-based systems, *ACM Comput. Surv.*  
412 54 (2021).
- 413 [17] I. A. Khan, N. Moustafa, D. Pi, W. Haider, B. Li, A. Jolfaei, An enhanced multi-stage deep learning framework for detecting malicious  
414 activities from autonomous vehicles, *IEEE Transactions on Intelligent Transportation Systems* (2021) 1–10.
- 415 [18] H.-H. Jebamikyous, R. Kashef, Autonomous vehicles perception (avp) using deep learning: Modeling, assessment, and challenges, *IEEE*  
416 *Access* 10 (2022) 10523–10535.
- 417 [19] Q. Zhu, W. Li, H. Kim, Y. Xiang, K. Wardega, Z. Wang, Y. Wang, H. Liang, C. Huang, J. Fan, H. Choi, Know the unknowns: Ad-  
418 dressing disturbances and uncertainties in autonomous systems, in: *Proceedings of the 39th International Conference on Computer-Aided*  
419 *Design*, Association for Computing Machinery, New York, NY, USA, 2020. URL: <https://doi.org/10.1145/3400302.3415768>.  
420 doi:doi:10.1145/3400302.3415768.
- 421 [20] N. M. Ben Lakkhal, L. Adouane, O. Nasri, J. Ben Hadj Slama, Safe and adaptive autonomous navigation under uncertainty based on sequential  
422 waypoints and reachability analysis, *Robotics and Autonomous Systems* 152 (2022) 104065.
- 423 [21] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, M. Kwiatkowska, Uncertainty quantification with statistical guarantees in end-  
424 end autonomous driving control, 2020 *IEEE International Conference on Robotics and Automation (ICRA)* (2020).
- 425 [22] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: *Proceedings*  
426 *of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2017, p.  
427 6405–6416.
- 428 [23] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, Y. L. Traon, Test selection for deep learning systems, *ACM Trans. Softw. Eng. Methodol.* 30  
429 (2021).
- 430 [24] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, *Commun. ACM* 62 (2019) 137–145.
- 431 [25] Y. Tian, K. Pei, S. Jana, B. Ray, Deeptest: Automated testing of deep-neural-network-driven autonomous cars, in: *Proceedings of the 40th*  
432 *international conference on software engineering*, 2018, pp. 303–314.
- 433 [26] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, et al., Deepmutation: Mutation testing of deep learning  
434 systems, in: *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2018, pp. 100–111.
- 435 [27] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, J. Zhao, Deepct: Tomographic combinatorial testing for deep learning systems, in: *2019*  
436 *IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2019, pp. 614–618.
- 437 [28] X. Xie, L. Ma, F. Juefei-Xu, H. Chen, M. Xue, B. Li, Y. Liu, J. Zhao, J. Yin, S. See, Deephunter: Hunting deep neural network defects via  
438 coverage-guided fuzzing, *arXiv preprint arXiv:1809.01266* (2018).
- 439 [29] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, Y. L. Traon, Test selection for deep learning systems, *ACM Trans. Softw. Eng. Methodol.* 30  
440 (2021).
- 441 [30] Y. Djenouri, A. Belhadi, G. Srivastava, D. Djenouri, J. C.-W. Lin, Vehicle detection using improved region convolution neural network for  
442 accident prevention in smart roads, *Pattern Recognition Letters* 158 (2022) 42–47.
- 443 [31] M. Mekala, W. Park, G. Dhiman, G. Srivastava, J. H. Park, H.-Y. Jung, Deep learning inspired object consolidation approaches using lidar  
444 data for autonomous driving: a review, *Archives of Computational Methods in Engineering* (2021) 1–21.
- 445 [32] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, *Machine*  
446 *Learning* 110 (2021) 457–506.
- 447 [33] A. Gane, T. Hazan, T. Jaakkola, Learning with Maximum A-Posteriori Perturbation Models, in: S. Kaski, J. Corander (Eds.), *Proceedings of*  
448 *the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*,  
449 PMLR, Reykjavik, Iceland, 2014, pp. 247–256. URL: <https://proceedings.mlr.press/v33/gane14.html>.
- 450 [34] W. Maddox, T. Garipov, P. Izmailov, D. P. Vetrov, A. G. Wilson, A simple baseline for bayesian uncertainty in deep learning, *CoRR*  
451 *abs/1902.02476* (2019).
- 452 [35] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii,  
453 in: *International conference on parallel problem solving from nature*, Springer, 2000, pp. 849–858.
- 454 [36] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah,  
455 A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, V. Shet, Level 5 perception dataset 2020, [https://level-5.global/level5/  
456 data/](https://level-5.global/level5/data/), 2019.
- 457 [37] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: *Proceedings of the*  
458 *IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

## 7 Towards robust autonomous driving systems through adversarial 8 test set generation

9 Devrim Unal<sup>a,\*</sup>, Ferhat Ozgur Catak<sup>b</sup>, Mohammad Talal Houkan<sup>c</sup>, Mohammed Mudassir<sup>c</sup> and  
10 Mohammad Hammoudeh<sup>d</sup>

11 <sup>a</sup>organization=KINDI Center For Computing Research, Qatar University, city=Doha, citysep=, postcode=2713, country=Qatar

12 <sup>b</sup>organization=Electrical Engineering and Computer Science Department, University of Stavanger, city=Rogaland, citysep=, postcode=4021,  
13 country=Norway

14 <sup>c</sup>organization=Department of Mechanical and Industrial Engineering, Qatar University, city=Doha, citysep=, postcode=2713, country=Qatar

15 <sup>d</sup>organization=Information and Computer Science Department, King Fahd University of Petroleum and Minerals, city=Dhahran, citysep=,  
16 postcode=31261, country=Saudi Arabia

17

### 18 ARTICLE INFO

19

#### 20 Keywords:

21 risk-aware autonomous systems

22 DL

23 test set generation

24 uncertainty

25

26

27

28

29

30

31

32

33

### ABSTRACT

Correct environmental perception of objects on the road is vital for the safety of autonomous driving. Making appropriate decisions by the autonomous driving algorithm could be hindered by data perturbations and more recently, by adversarial attacks. We propose an adversarial test input generation approach based on uncertainty to make the machine learning (ML) model more robust against data perturbations and adversarial attacks. Adversarial attacks and uncertain inputs can affect the ML model's performance, which can have severe consequences such as the misclassification of objects on the road by autonomous vehicles, leading to incorrect decision-making. We show that we can obtain more robust ML models for autonomous driving by making a dataset that includes highly-uncertain adversarial test inputs during the re-training phase. We demonstrate an improvement in the accuracy of the robust model by more than 12%, with a notable drop in the uncertainty of the decisions returned by the model. We believe our approach will assist in further developing risk-aware autonomous systems.

<sup>\*</sup> dunal@qu.edu.qa (D. Unal); f.ozgur.catak@uis.no (F.O. Catak); mh1208170@qu.edu.qa (M.T. Houkan); mh1204033@qu.edu.qa (M. Mudassir); M.Hammoudeh@kfupm.edu.sa (M. Hammoudeh)

ORCID(s): 0000-0003-3146-3502 (D. Unal); 0000-0002-2434-9966 (F.O. Catak); 0000-0002-6491-4819 (M.T. Houkan); 0000-0002-0093-1257 (M. Mudassir); 0000-0003-1058-0996 (M. Hammoudeh)

**Conflict of interest statement**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Journal Pre-proof