# Autonomous driving quality assurance with data uncertainty analysis

1st Tinghui Ouyang
*Digital Architecture Research Center (DigiARC)*
*National Institute of Advanced Industrial Science and Technology (AIST)*
Tokyo, Japan
ouyang.tinghui@aist.go.jp

2nd Yoshinao Isobe
*Cyber Physical Security Research Center*
*AIST*
Osaka, Japan
y-isobe@aist.go.jp

3rd Saima Sultana
*DigiARC*
*AIST*
Tokyo, Japan
saima.sultana@aist.go.jp

4th Yoshiki Seo
*DigiARC*
*AIST*
Tokyo, Japan
y.seo@aist.go.jp

5th Yutaka Oiwa
*DigiARC*
*AIST*
Tokyo, Japan
y.oiwa@aist.go.jp

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9891977

***Abstract*—Deep Learning (DL) based self-driving systems are vigorously developing in big companies. While, as several serious accidents were reported with life- and property-loss, the issue of robustness in DL-based self-driving systems inspires great attention, especially facing with some high-risk cases, like adversarial inputs or corner case scenarios in driving. Considering the existing methods are cumbersome in real driving, therefore this paper proposed a novel and simple way which studies data's uncertainty to rise alarm for manual checking. This method developed a metric describing corner case with respect to DL models, and subsequently evaluated data uncertainty. Experiments on a self-driving system verified the feasibility and usefulness of the proposed method. Code in this paper is released [1].**

***Index Terms*—self-driving, deep learning, data uncertainty, robustness, corner case data**

## I. Introduction

Automation is widely regarded to play the most important part in automotive industries, which brings great benefits to humans' living modes, environment and economy [2]. Moreover, as the rapid development of Artificial Intelligence (AI), especially Deep Learning (DL) based technologies, it makes the self-driving come true from the science fiction previously [3]. Nowadays, many big companies have been competing in the area of DL-based autonomous driving, like Tesla, Ford, Waymo, Google and etc. [4]- [5]. For example, Google's self-driving car is famous because its accident-free record has been accumulated thousands of miles [6]. This situation leads several countries to prepare to adjust their laws to permit autonomous driving cars running on public streets in the near future [7].

While, as we know, the robustness and stability are always the vital issue about DL quality assurance [8]. Due to the complex structure and a huge amount of parameters within DL models, sometimes a small perturbation or a little noise added into inputs might cause DL make a completely wrong decision. For example, it is commonly known in image recognition that only one pixel change in an image could let a high-accuracy DNN model recognize it as a different class [9]. Moreover, in self-driving area, the robustness of DL models even directly involving safety, the wrong performance of DL-based self-driving system would directly lead to life- and property-loss. Many real-world collision cases have been reported since developing DL-based autonomous driving system. For example, In [10], a collision between a Tesla car and a trailer was reported since the Tesla's DL system failed to deal with scenarios of "white color against a brightly lit sky" and the "high ride height". Another collision was reported on a Google self-driving car crashed to a bus due to DL's wrong decision [11]. Other cases were also reported, like self-driving cars could also not correctly deal with emergency [12]- [13]. All these cases seem to tell us that the type I error [14], namely False Positive (FP) events, cannot be avoided in DL-based systems, even for companies like Tesla and Google having strong strength on economy and scientific technologies, but they also mightily inspires scholars' enthusiasms on studying the robustness and stability of DL models.

Currently, most of AI robustness analysis is based on adversarial testing data, and two major directions can be summarized, such as robustness evaluation and robustness improvement. To implement these robustness analysis, some effective attack algorithms were generally proposed firstly, then used to generate adversarial samples with respect to the original testing data, for example, the commonly-used FGSM, BIM, C&W, JSMA algorithms [15]- [17]. Moreover, some other methods were also reported in literature, like DeepFool [18], CLEVER [19] and etc. Then, aiming at the first topic, robustness of the given DL models can be measured on the basis of the comparison between the adversarial testing dataset and the corresponding original one. Generally, robustness measurement is calculated as the $L_p$ norm of this distortion. The other research topic is to make use of generated adversarial samples for AI robustness improvement. This is realized by adding adversarial data with corrected labels into the original training dataset for model retraining. In this way, the retrained models can be adversarially robust to those attacks used for adversarial samples generation, but it may still be vulnerable to unseen attacks [20].

From the above summaries on AI robustness, there are still some shortages hidden in AIs' quality assurance. As it mentioned, one shortage is that the retrained DL model is still vulnerable to unseen attacks, since the adversarial generation is tightly related to model. The other shortage is the boundary samples. When AI systems are attacked by a small bug, but not discovered by humans, wrong decision is highly possible to be made. Moreover, there exists one possible scenario that also lead DL models to wrong decisions. For example, when all circumstances look normal in autonomous driving, but the AI system makes a dangerous decision on turning. The reason behind it is to a great extent because no enough data of this case are considered in model training process. Like these samples possibly causing incorrect or unexpected behaviors, we can call them as corner case data [21], including those incorrectly recognized, boundary and high-risk samples. There were also many researches considering corner case data for AI model improvement, e.g. DeepXplore [22] and DeepRoad [23] retrained DL models with generated corner cases. However, it should be aware that it is impossible to generate all corner cases for model improvement.

Therefore, a way studying testing data's uncertainty with respect to DL models seems more feasible for safe AI operation. This type of methods is actually to using data uncertainty to evaluate the confidence of data w.r.t. model's right decision, namely to detect those high-risk data (adversarial and corner case data) before the final decision-making. In this way, most of FP events could be separated for manual checking, therefore the final performance of AI models will be improved. There are also many papers reporting this method. For example, an extra 'detector' network was trained specifically for adversarial data detection in [24]. In [25], feature squeezing was firstly applied to generate multiple inputs. Then, determining an uncertain sample when multiple inputs' predictions are inconsistent. Similarly, in [26] data uncertainty is determined by combining results of multiple predictions which are based on different dropout variance of model. In summary, it is seen that, ways for uncertain data detection generally require multiple running or extra model support, they have cumbersome procedures and high overhead in real operation, e.g. in autonomous driving.

Then, we may think about "is it possible to propose a simple way on the basis of the studied DL model itself?" Aiming at this purpose, this paper proposes a novel and simple way for uncertain data detection. This method proposes to describe all corner case data firstly, including adversarial data. Then, developing a simple and useful metric evaluating data's uncertainty with respect to the given DNN model, and implementing uncertain data detection in the testing process. Moreover, we will apply for this proposed method on the basis of the autonomous driving system, and evaluate its performance. The paper will be organized as follows. Section 2 will discuss the motivation of data uncertainty of DL systems. Section 3 will introduce how to implement the data uncertainty analysis on a self-driving system. Section 4 will implement experiments and discussion. Section 5 will conclude the research in this paper.

## II. Motivation of data uncertainty study

In AI quality assurance, wrong warning is generally allowable, sometimes even welcome, but wrong decision is absolutely not acceptable! This is also the motivation of why we need to study data's uncertainty and to detect the warning points. Actually, studying data's uncertainty is namely data dependability/reliability analysis. Dependability/reliability is a useful indicator in software quality analysis [27]. It generally contains two parts: one is model reliability, the other one is testing data reliability. In AI quality assurance, the robustness/stability of model describes the similar meaning of model reliability. Data reliability can be described by data uncertainty, tightly related to corner cases (bugs).

Traditional software is generally developed by programming language and abides by some logical criterion. No software can perform perfectly, they generally have limitations (problem domain) and bugs. The bugs can be eliminated through the logic detection by developers. However, for machine learning based software, they are developed by data training and model architecture, different with traditional software. Their bugs are certainly related to data and AI algorithms. For algorithm and architecture bugs, it is easy to eliminate them which usually lead to low accuracy. For data related bugs, namely to find the corner cases in AI operation. There are a lot of literature studying on finding corner cases (bugs) in deep learning models, like DeepXplore, DeepRoad and so on [22]-[23]. The common conclusion is the amount of corner case data is infinite, we can just propose different techniques to find as many corner cases (bugs) as possible.

While, is it truly necessary to find out all bugs for both traditional software and DL-based software? As the description above, no software can perform perfectly at any circumstances, excluding all bugs is an ideal expectation but not realistic. For traditional software, we can exclude many bugs via logic detection, and also constrain inputs in the logical problem domain to eliminate bug occurrence. For DL-based software, the input domain can be complex, possibly Euclidean and non-Euclidean. It is true to define the problem domain of DL-based software via logic description. However, this problem domain still has a gap to directly affect DL-based software's performance. Therefore, studying a way avoiding bugs seems more feasible than finding and defining all possible bugs, especially in DL-based software engineering. This also explains the motivation of why we need to study data uncertainty, or data dependability in DL systems.

## III. Data uncertainty analysis in a self-driving system

As the introduction above, in this paper we aim at studying data uncertainty/dependability on self-driving, namely to

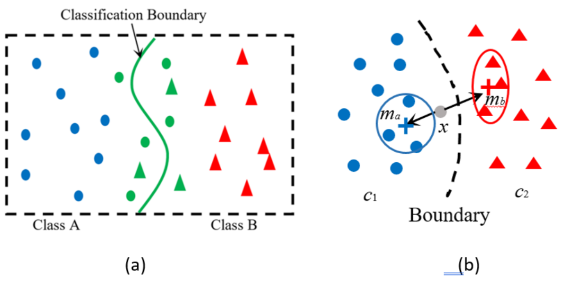

Fig. 1. Diagram of corner case data and their descriptors. (a) corner case data; (b) distanced-based descriptors

propose a metric for data uncertainty for guiding self-driving. To realize this purpose, we plan to implement the following studies, such as uncertain data description, i.e. corner case data, data uncertainty calculation, and application in self-driving systems.

## A. Corner case description

As it mentioned that corner case data means data leading a DL system to incorrect or unexpected behaviors, so corner case data is uncertain with respect to DL models. Then, on the basis of its concept, corner case data should contain both the erroneous samples and adversarial ones with respect to a specific DL model. However, different DL models may have different performance on specific samples. Moreover, DL models may be attacked by bugs in the operation process, but not aware for humans. In this way, boundary samples close to the decision-making hyper-plane are also with high risk to cause unexpected behaviors. Therefore, corner case domain should be more general, not with respect to a specific model. In this way, we describe the corner case data domain as shown in Figure 1(a). Similar with the execution of traditional software engineering, corner case for AI quality assurance could contain erroneous data and high-risk data close to the decision-making boundary [28]- [29]. In Figure 1(a), two classes of data are colored as blue and red; corner case data is colored as green, including erroneous data and data close to classification boundary. To describe this type of data, here we refer to the idea of surprise adequacy in [30] which describes the novelty between testing data and training data. However, different with data surprise description, corner case description should consider data distribution in the decision-making space as well. Therefore, in this paper we proposed a modified descriptor based on the distance-based surprise adequacy (DSA) [31], as presented in Figure 1(b). In this idea, corner case data is described as points away from its belonging category in classification problem, therefore we can define the corner case descriptor $D_{cc}$ as the ratio of distance to its own class and that to its nearest class, as the following form

$$D_{cc} = \frac{dist_a}{dist_b} \tag{1}$$

where, $dist$ is defined as the distance of testing data $x$ to a given class, and calculated as below

$$\left\{ \begin{array}{l} dist_a = ||x - m_a|| \\ dist_b = ||x - m_b|| \end{array} \right. \tag{2}$$

Here, $a$ and $b$ represent the testing data's belonging class and its nearest class respectively. Meanwhile, the distance to a class can be reflected by the center ($m$) of local/global descriptors of classes. For example, the local descriptor of a class is presented by the nearest neighbor, in this way the corner case descriptor can have good capability on describing corner case data, avoiding the influence caused by extreme individual points.

## B. Data uncertainty description

It is clear that, corner case inputs have large uncertainty to lead DL to make wrong decision. That means corner case data itself can describe data uncertainty. Therefore, before the study of data uncertainty and AI robustness analysis, the first task can be describing corner case in numeric way. In the above description, a corner case descriptor is defined based on DSA, so can we use the original testing data to calculate the descriptor's value directly?

As we know, image data are too complex to reflect its distribution relation to the decision-making boundary directly. In [30], the behaviors of data with respect to a given DL model are described by the activation status in specific hidden layers. That is because DNN models have a well-known feature of transforming non-linear data into linear separable data via deep learning. Therefore, the activation status of data in hidden layers are useful to capture data's behaviors, especially with respect to the final decision. As the definition in [30], we also define behaviors of data with respect to neurons of a DL system as activation trace (AT).

By assuming a DL system having a set of neurons $\boldsymbol{N} = \{n_1, n_2, \ldots\}$, the activation trace of a given data sample $x$ with respect to a neuron $n$ is denoted as $a_n(x)$. For an ordered subset of neurons $N \subseteq \boldsymbol{N}$, the activation trace of data $x$ is $a_N(x)$ whose cardinality is equal to $|N|$. Similarly, assuming a testing set as $X = \{x_1, x_2, \ldots\}$, its activation trace over neurons in $N$ can be expressed as the following form

$$A_N(X) = \{a_N(x) | x \in X\} \tag{3}$$

It is noted that the activation trace of the testing data $X$ is trivially available after each execution of the DL systems since AIs are driven by data instead of control-flow in traditional software engineering. Moreover, through the execution of DL systems on data $X$, we can capture both DL's behaviors and the difference of data on activation traces. The latter will be useful in data description. Therefore, the elements $x$ and $m$ in equation (2) can be replaced by $a_N(x)$ and $a_N(m)$ in corner case description, where $N$ can be selected as neurons of a specific hidden layer. Then, the uncertainty of testing data can be calculated as the value of $D_{cc}$ directly.

### C. An example self-driving system

In the paper, the application system for data uncertainty study is based on a self-driving system, for example, the famous project DAVE2 developed by NVIDA is applied here [32]. This system is based on the general convolution neural networks (CNN) architecture, and based on the input image for steering angle prediction, as shown in Figure 2. It consists of 9 layers, including a normalization layer, 5 convolution layers and 3 full-connection layers.

The normalization layer is to pre-process the input images from an actual driving circumstance. Those convolution layers are applied to learn important image patterns in self-driving. Among them, the first three layers are all designed with $2 * 2$ stride and $5 * 5$ kernel. The last two layers are without stride, and with 3$*$3 kernel in construction. The final 3 full-connection layers are designed to realize the given application requirement, for example turning direction recognition, steering angle prediction in [32]. According to the above analysis, to extract the most useful activation trace for data uncertainty study, we can choose the final full-connection layer (marked as red in Figure 2, and taking its activation trace for corner case detection and data uncertainty analysis as described above.

## IV. Experiments and analysis

### A. Self-driving data description

As the description in self-driving system NVIDA-DAVE2, the data was collected from New Jersey, as well as highway data from Illinois, Michigan, Pennsylvania, and New York. Moreover, data are generally collected from three front cameras (left, central, right) when car runs under several weather conditions. However, for the presented CNN model for self-driving above, only the images from the central camera are considered as inputs, and the output is just for steering angle control. In our study, we can use the pre-trained model provided by the DAVE2 project [32], or use the available data for training a new model. The given dataset contains totally 45,568 images, among which there are 45,405 labeled images and 162 unlabeled images. In our research, we can take divide labeled data into training and testing set with a number of 40,000 and 5,406 images respectively.

### B. Model for steering control direction prediction

According to the available pretrained model, we can implement the prediction of steering angle in self-driving. Figure 3 shows the predicted angles vs. the originally measured angles. It is seen that in most of cases prediction is consist with observation, but there are still a lot of cases having large difference between predicted and measured steering angles. While, here we may have a trouble on evaluating model's performance with this angle difference, since there exists a transformation between wheel angle and steering wheel angle. For example, a general automatic car may have one degree of wheel angle change with 12 degrees of steering angle change, so a criterion should be given to evaluate the steering angle prediction. Moreover, the corner case description in section 3 is mainly based on classification problem, but here it may not be applicable. Aiming at these two issues, we propose to modify the original self-driving system for the target of turning direction prediction, as shown in Figure 4.

In Figure 4, we can see the modification is just to change the output from steering angle to steering direction prediction. In this research, we can simply define three kinds of turning direction, such as going straight (-$45^o$  $45^o$), turning left ($< -45^o$) and turning right ($> 45^o$). Then, we can label the training data and train a model for steering control direction prediction.

### C. Corner case detection and data uncertainty analysis

Based on the modified DL model for self-driving, we can further calculate the values of $D_{cc}$ on a specific hidden layer, e.g. the final FC layer marked red in Figure 4. Then, the probability distribution of all $D_{cc}$ values are presented in Figure 5(a). Moreover, to evaluate the feasibility of $D_{cc}$ on corner case data description, we can consider to utilize $D_{cc}$ values for corner case data detection directly. However, considering corner case generally contains boundary, adversarial and high-risk points, so we can only consider part corner case data in evaluation, namely the misclassified points in steering control direction prediction, which are easily recognized for a DL model. Then, the ROC curve about using $D_{cc}$ values for misclassification data recognition is plotted, as shown in Figure 5(b) where TPR and FPR represent the values of true positive rate and false positive rate respectively. The definitions of these metrics are denoted as below

$$\begin{cases} TPR = \frac{card(TP)}{card(TP)+card(FN)} \\ FPR = \frac{card(FP)}{card(FP)+card(TN)} \end{cases} \quad (4)$$

where, $TP/FP/FN/TN$ represents the true positive/false positive/false negative/true negative data in recognition; the function $card(*)$ is to calculate the number of the given data.

From the results of Figure 5, some interesting phenomenon can be discovered. First, according to Figure 5(a), it is seen that most of data has low values of $D_{cc}$. In Figure 5(b), the AUC [33]-ROC value of using $D_{cc}$ for incorrectly classified data detection reaches 0.9613, which implies a good performance on misclassification recognition by using $D_{cc}$, and it also illustrates that $D_{cc}$ values have good relationship with misclassification. Combined with the results of Figure 5(a), we can conclude that $D_{cc}$ values are related with corner case data to some extent, and useful to evaluate the uncertainty of data further.

Subsequently, to evaluate testing data's uncertainty based on $D_{cc}$ values, we need to determine a suitable threshold for discriminating certainty data and uncertain data. For certain data, we can accept the decision made by DL models. For the uncertain data, manual checking would be needed on the basis of DL models' recognition. To implement this selection, we can select different values of $D_{cc}$ threshold, and calculate the testing accuracy of certain data. Then, data uncertainty and model's performance can be analyzed based on suitable threshold, as shown in Figure 6.

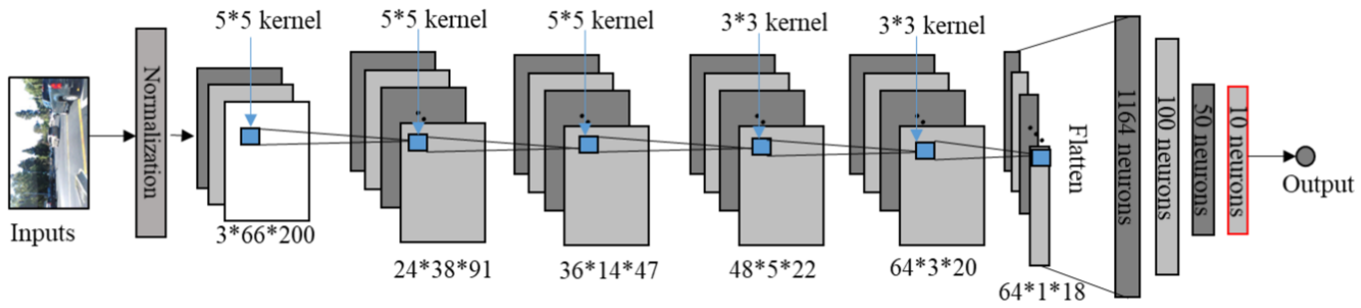

Fig. 2. Architecture of DL-based self-driving model

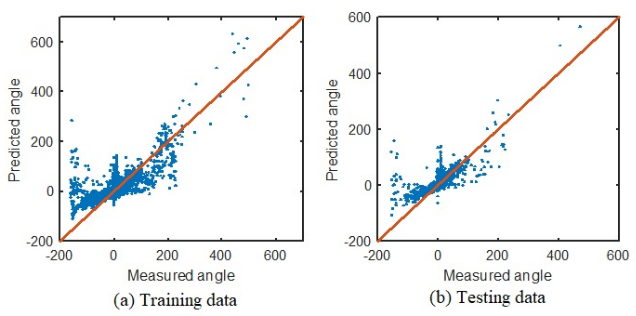

Fig. 3. Predicted steering angle vs. measured ones.

In Figure 6(a), we can see that as the threshold of $D_{cc}$ decreases, the testing accuracy of certain data can be increased. Meanwhile, to reflect the ratio of certain data and uncertain data, Figure 6(b) plots the cumulative probability curve as the variance of $D_{cc}$ threshold. For example, if we set the $D_{cc}$ threshold as 0.5129, the testing accuracy of certain data can increase from $96.07\%$ to $99.61\%$, which will be satisfied results for self-driving. Meanwhile, the percentage of uncertain data is $(1-0.8778)*100\% = 12.22\%$, implying $12.22\%$ of data will need manual checking on the basis of DLs' decision. In this way, we can add a "warning" sign to this kind of data. Pictures in Figure 7 give out some real examples in the given self-driving system, showing how the new self-driving system performs with data uncertainty analysis.

In Figure 7, two real cases (case 42210 and case 42230) from the testing data are selected. First, it is easy to know that the driving car is moving and going to turn left currently from Figure 7. However, based on the self-driving system, they are required to turn right with a steering angle of $61.64^o$ and $111.50^o$ respectively. These decisions made by AI are absolutely wrong, and will cause serious accident like crash to the black pickup truck, therefore we must discard the DL model's decision at this time. In the original model, the self-driving system cannot do that. If introducing the data uncertainty analysis, e.g. setting $D_{cc}$ threshold as 0.5129 as the uncertainty criterion, then we can recognize these two cases as uncertain, and marked as warning. Finally, humans intervention is introduced for these uncertain cases to avoid serious accident.

### D. Testing on more high-risk cases

Moreover, in order to study the generalization ability of this proposed method on data uncertainty analysis, we implement more examples on testing data with attacks, namely adding some perturbations into the original self-driving images. In this paper, we implement three kinds of attacks, such as the light change, occlusion addition, and noise perturbation, as shown in Figure 8.

Based on these testing data with attacks, we can calculate their $D_{cc}$ values based on the proposed method above, and use them to study the relation between Dcc values and misclassification, as shown in Figure 9.

From Figure 9, we can also conclude that $D_{cc}$ values of these data have tight relation with misclassification which testing data is attacked, since the AUC-ROC values of these testing data with three different attacks all have large values. This also implies that $D_{cc}$ can also reflect the uncertainty of attacked data. Furthermore, assuming we also set the uncertainty threshold as 0.5129, we can calculate the testing accuracy of certain data in each attack. For the light change, the testing accuracy can increase from $68.90\%$ to $98.81\%$. For the addition of occlusion, the testing accuracy can increase from $78.74\%$ to $98.80\%$. For the noise perturbation, the testing accuracy can increase from $77.35\%$ to $98.72\%$. It is seen that when using a $D_{cc}$ threshold for data uncertainty analysis and letting uncertain data for manual checking, the testing accuracy of certain data in DL-based self-driving can be greatly increased even when the testing data is attacked. In a conclusion, we can see the proposed way for data uncertainty is generalized and useful to attacked data. If we introduce it into the DL-based self-driving systems, self-driving would be more reliable and safe to reduce serious accident in the operation.

## V. Conclusions

In this paper, to improve the safety in DL-based self-driving systems, we proposed to study the input data's uncertainty, and leave uncertain data for manual checking. This research proposed to utilize the activation status of DL models for corner case data description, which is verified useful to detect

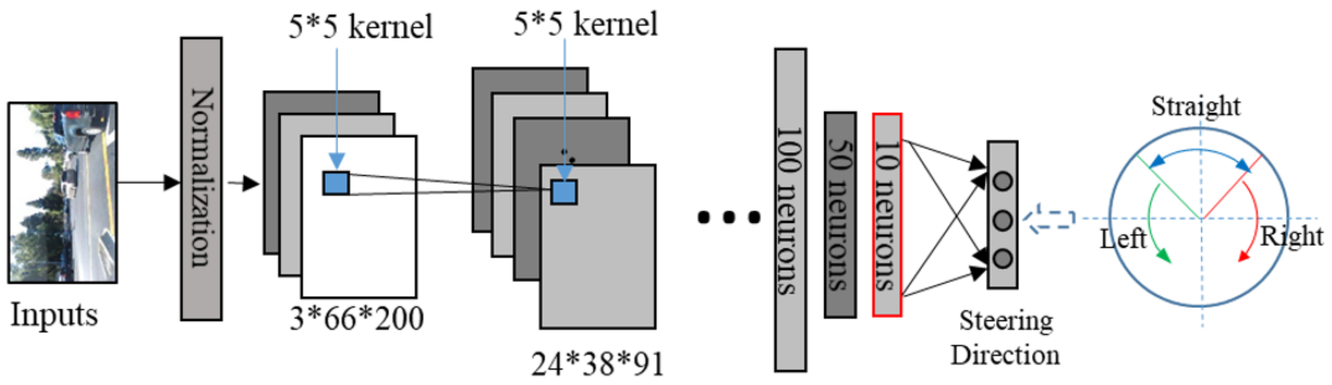

Fig. 4. Modification of self-driving model for steering control direction prediction

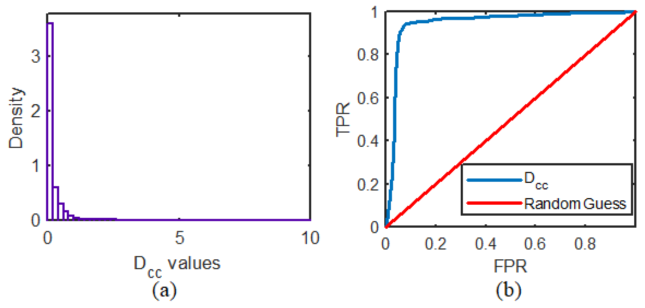

Fig. 5. (a)distribution of $D_{cc}$ values; (b) ROC curve by using $D_{cc}$ for misclassification recognition;

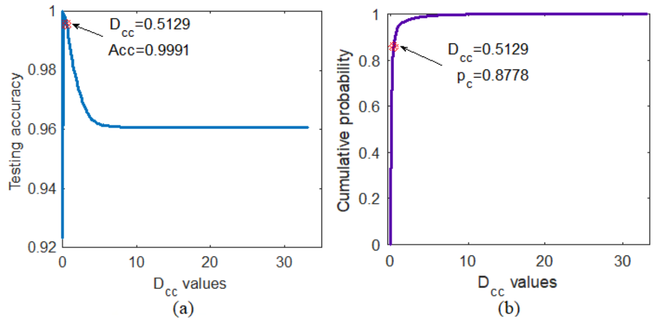

Fig. 6. (a) Different Dcc threshold vs. testing accuracy; (b) different Dcc threshold vs cumulative probability.

misclassified data. Then, a corner case data descriptor $D_{cc}$ was constructed, and used to analyze data's uncertainty. With the consideration of data uncertainty analysis, accuracy of the given self-driving system (DAVE2) with certain confidence can be improved greatly on steering direction recognition. While, currently the proposed metric for data uncertainty analysis is simple, better metrics can be studied in the future. Moreover, in this research the self-driving system is simply considered for steering control direction, more real issues should be considered in industrial self-driving systems.

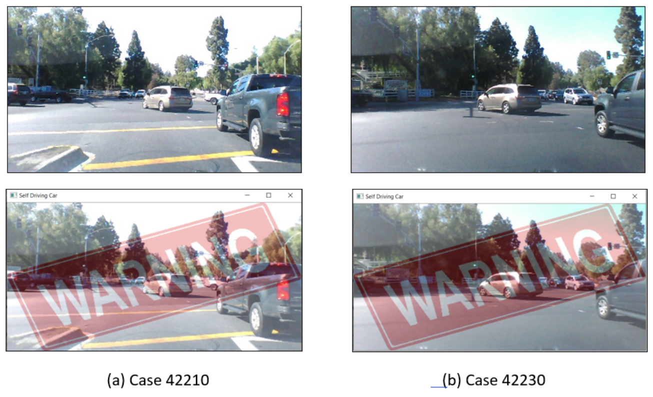

Fig. 7. Self-driving system with data uncertainty analysis

## VI. Acknowledgments

This research is supported by the project 'JPNP20006', commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

## References

[1] https://github.com/thouyang/self_driving_with_corner_cases
[2] Diels, C. Bos, J.E. (2015). Self-Driving Carsickness. Applied Ergonomics, (53) Part B, pp. 374-382.
[3] Rao, Q., and Frtunikj, J. (2018, May). Deep learning for self-driving cars: Chances and challenges. In Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems (pp. 35-38).
[4] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., and Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2446-2454).
[5] Cerf, V. G. (2018). A comprehensive self-driving car test. Communications of the ACM, 61(2), 7-7.
[6] BBC news (2013). Driverless cars to be tested on UK roads by end of 2013. http://www.bbc.co.uk/news/technology-23330681 [29 October 2014].
[7] Greenblatt, N. A. (2016). Self-driving cars and the law. IEEE spectrum, 53(2), 46-51.
[8] Zhang, J. M., Harman, M., Ma, L., and Liu, Y. Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering, 2020.

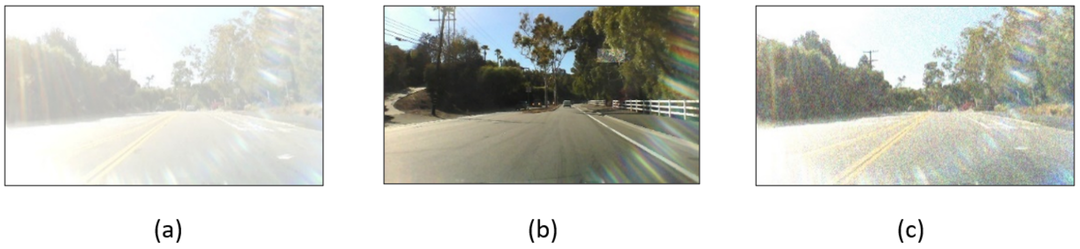

Fig. 8. Examples of generated adversarial testing data. (a) Light change (going straight → turning left); (b) occlusion (going straight → turning right); (c) noise (going straight → turning right);

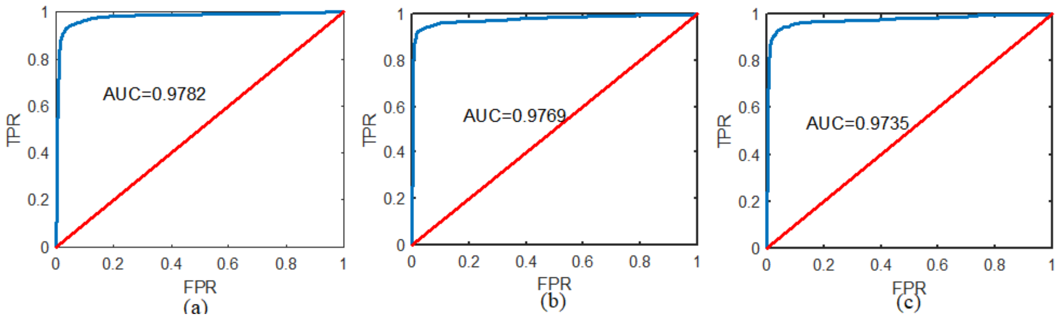

Fig. 9. AUC-ROC of using Dcc values for misclassification recognition. (a) light change; (b) occlusion; (c) noise;

[9] Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation, 23(5), 828-841.

[10] tesla-accident 2016. "Understanding the fatal Tesla accident on Autopilot and the NHTSA probe," https://electrek.co/2016/07/01/

[11] google-accident 2016. "A Google self-driving car caused a crash for the first time," http://www.theverge.com/2016/2/29/11134344/googleselfdriving-car-crash-report.

[12] https://www.abc15.com/news/region-southeast-valley/chandler/waymo-car-involved-in-chandler-arizona-crash

[13] https://www.ft.com/content/89692fee-1181-11e7-80f4-13e067d5072c

[14] Von der Malsburg, T., and Angele, B. (2017). False positives and other statistical errors in standard analyses of eye movements in reading. Journal of memory and language, 94, 119-133.

[15] Kannan, H., Kurakin, A., and Goodfellow, I. Adversarial logit pairing. arXiv preprint arXiv:1803.06373, 2018.

[16] Carlini, N., and Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp) (pp. 39-57). IEEE.

[17] Papernot, N., Mcdaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (2016), IEEE, pp. 372–387.

[18] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. arXiv preprint arXiv:1511.04599, 2015.

[19] Weng, T. W., Zhang, H., Chen, P. Y., Yi, J., Su, D., Gao, Y., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. arXiv preprint arXiv:1801.10578, 2018.

[20] Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204, 2017.

[21] Tian, Y., Pei, K., Jana, S., and Ray, B. (2018, May). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th international conference on software engineering (pp. 303-314).

[22] Pei, K., Cao, Y., Yang, J., and Jana, S. (2017, October). Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).

[23] Zhang, M., Zhang, Y., Zhang, L., Liu, C., and Khurshid, S. (2018, September). DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 132-142). IEEE.

[24] Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017). On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267.

[25] Xu, W., Evans, D., and Qi, Y. (2017). Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155.

[26] Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. (2017). Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410.

[27] Kozik, R., Choraś, M., Puchalski, D., and Renk, R. (2018, July). Platform for software quality and dependability data analysis. In International Conference on Dependability and Complex Systems (pp. 306-315). Springer, Cham.

[28] Ouyang, T., Marco, V. S., Isobe, Y., Asoh, H., Oiwa, Y., and Seo, Y. (2021, May). Corner case data description and detection. In 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN) (pp. 19-26). IEEE.

[29] Ouyang, T., Isobe, Y., Marco, V. S., Ogata, J., Seo, Y., and Oiwa, Y. (2021, August). AI robustness analysis with consideration of corner cases. In 2021 IEEE International Conference on Artificial Intelligence Testing (AITest) (pp. 29-36). IEEE.

[30] Kim, J., Feldt, R., and Yoo, S. (2019, May). Guiding deep learning system testing using surprise adequacy. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 1039-1049). IEEE.

[31] Ouyang, T., Marco, V. S., Isobe, Y., Asoh, H., Oiwa, Y., and Seo, Y. (2021). Improved Surprise Adequacy Tools for Corner Case Data Description and Detection. Applied Sciences, 11(15), 6826.

[32] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... and Zieba, K. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

[33] Narkhede, S. (2018). Understanding auc-roc curve. Towards Data Science, 26, 220-227.