2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC) | 979-8-3503-7696-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/COMPSAC61105.2024.00148

Enhancing Valid Test Input Generation with Distribution Awareness for Deep Neural Networks

Jingyu Zhang[1], Jacky Keung[1], Xiaoxue Ma[1], Xiangyu Li[2], Yan Xiao[3], Yishu Li[1], and Wing Kwong Chan[1]
[1]Department of Computer Science, City University of Hong Kong, Hong Kong, China,
{jzhang2297-c, xiaoxuema3-c, yishuli5-c}@my.cityu.edu.hk
{jacky.keung, wkchan}@cityu.edu.hk
[2] Department of Electrical and Computer Engineering, McGill University, Montreal, Canada,
xiangyu.li3@mail.mcgill.ca
[3] School of Cyber Science and Technology, Shenzhen Campus of Sun Yat-sen University, Shenzhen, China,
xiaoy367@mail.sysu.edu.cn

***Abstract*—Comprehensive testing is important in improving the reliability of Deep Learning (DL)-based systems. Various Test Input Generators (TIGs) have been proposed to generate misbehavior-inducing test inputs. However, the lack of validity checking in TIGs often results in the generation of invalid inputs (i.e., out of the learned distribution), leading to unreliable testing. To save the effort of manually checking the validity and improve test efficiency, it is important to assess the effectiveness and reliability of automated validators.**

In this study, we comprehensively assess four automated Input Validators (IVs). Our findings show that the accuracy of IVs ranges from 49% to 77%. Distance-based IVs generally outperform reconstruction-based and density-based IVs for both classification and regression tasks.

Based on the findings, we enhance existing testing frameworks by incorporating distribution awareness through joint optimization. The results demonstrate our framework leads to a 2% to 10% increase in the number of valid inputs, which establishes our method as an effective technique for valid test input generation.

***Index Terms*—Input Validation, Anomaly Detection, Deep Learning, Software Testing**

I. Introduction

With the rapid development of deep learning (DL), many software applications have integrated deep neural networks (DNNs) to improve their performance. Despite the notable achievements of DL-based models across various domains [1]–[3], research has indicated their vulnerability to malicious attacks, necessitating comprehensive testing [4]–[7]. Thorough testing of a DL system requires diverse and valid test inputs which can trigger model misbehavior. While plenty of Test Input Generators (TIGs) [4]–[6], [8]–[11] have been proposed to efficiently generate error-inducing inputs, most of them fall short in checking the validity of the generated test inputs.

Input validation is a fundamental concept in software testing. In traditional software engineering, developers must create valid test cases to effectively evaluate the functional logic of the software and ensure compliance with specifications. When an invalid test case is encountered, the software will handle the input with errors and exhibit unexpected behavior. In such cases, the functional logic cannot be adequately tested, rendering those test cases ineffective [12]–[14]. Generally, the enforcement built of input validation in software systems ensures only valid inputs are accepted for processing [13].

For DNNs, trained models are expected to generalize beyond the training data but within the valid input distribution [15]. However, DNNs often struggle to clearly distinguish between valid and invalid inputs, and they may still make predictions (possibly erroneous) for invalid inputs. For example, MNIST classifiers can predict digit labels for grayscale cat images with the same shape as MNIST samples, but these images are not meaningful in testing such classifiers. In DL testing, this brings difficulty in generating test inputs to effectively and correctly test the generalized behavior of a DNN. Unfortunately, research has demonstrated that existing TIGs may occasionally produce invalid inputs [7], [15]–[17].

To perform input validation for DNNs, expert labelling provides the most accurate but time-consuming assessment. To alleviate this burden, automated Input Validators (IVs) have been leveraged to automatically validate the input sample, sometimes referred to as anomaly detectors or out-of-distribution detectors [18], [19]. Different IVs identify invalid inputs (i.e., anomalies) through various methodologies. A straightforward method is to model the in-distribution density and classify out-of-distribution data as invalid, termed as density-based methods [20]–[23]. Another category, dubbed reconstruction-based methods, relies on the quality of the reconstructed image, such that hard-to-reconstruct samples are deemed invalid [24]–[28]. Distance-based methods calculate the feature-space distance [29]–[31] between the in-distribution (valid) and out-of-distribution (invalid) data.

A few testing techniques [7], [17] embedded an IV into their testing framework, aiming to automatically detect invalid inputs. Dola *et al.* [16] and Riccio *et al.* [15] examined the performance of reconstruction-based IVs and combined them with existing testing frameworks. Studying a single type of IV leads to two questions: *(1) Does the results generalize to other IVs? (2) Do we select the right combination of TIG and IV for the DNN models under test? (i.e., can the selected IV accurately detect invalid inputs)?* **Our solution:** We include the assessment of four IVs across three types

2836-3795/24/$31.00 ©2024 IEEE
DOI 10.1109/COMPSAC61105.2024.00148

(i.e., reconstruction-based, density-based, and distance-based) to evaluate the compatibility across all combinations of TIGs and IVs, and incorporate the best IV into the testing framework. The improvement in the number of valid inputs by our enhanced framework can reach up to 10% by human assessment.

We summarize the **key contributions** of this paper as follows:

(1) We systematically assess the effectiveness and reliability of four IVs with human-annotated validity labels. Our suggestion is to employ feature-based IVs on complex datasets. Compared with existing work, we further involve human evaluations for distance-based and density-based IVs.

(2) Building upon our empirical findings, we enhance the existing testing framework by incorporating distribution awareness through eight different joint optimizations. The results have demonstrated a 2% to 10% increase in the number of valid inputs by human assessment.

II. Background and Related Work

A. Overview of Input Validation

The typical workflow of test input validation, as shown in Figure 1, consists of two phases: 1) test input generation; 2) test input validation. By feeding seed inputs, TIGs can generate misbehavior-inducing test inputs. To effectively and efficiently reveal the generalization errors of DNNs under test, each generated test input should be assessed for validity, by either automated IVs or domain experts.

B. Existing TIGs

Diverse TIGs [4]–[7], [11] has been proposed to produce misbehavior-inducing inputs to the DL system under test. Some works [4], [5], [11] applied pixel-level perturbations to the original inputs based on the gradients computed with respect to a testing objective. Some [6], [32], [33] also conducted perturbation in the feature space, craving to generate more diverse inputs.

C. Existing IVs

An input is deemed *invalid* if it is anomalous or an outlier to the data distribution learned from training. Although an invalid input satisfies the acceptable input tensor shape of a DNN model, it might be semantically irrelevant to the given task. Due to the scarcity of representative anomalous data, a common practice [15], [16] is to apply semi-supervised learning, which utilizes the information from normal (valid) data only [18]. Semi-supervised techniques can be generally categorized into three types [19], [34]: statistical density-based, reconstruction-based, and distance-based methods. **Statistical Density-based Methods** explicitly estimate a statistical probabilistic model for valid data and identify low-density data as invalid. **Reconstruction-based Methods** rely on the reconstruction of images, which first maps the original input image to the latent space and then reconstructs the original input by finding an inverse mapping for the latent vector. A valid sample is expected to produce a good reconstruction with low reconstruction errors. **Distance-based Methods** classify a sample as invalid if the sample is located far away from the valid data in the feature space. To assess the validity of test inputs generated by testing techniques, Dola *et al.* [16], Zhang *et al.* [7], and *et al.* [17] utilized different types of IVs to validate the test inputs. Recently, Riccio *et al.* [15] conducted an empirical study to examine the effectiveness and reliability of reconstruction-based IVs, however, their results lack generalization to other types of IVs. We bridge the gap by performing a systematic effectiveness assessment for reconstruction-based, density-based, and distance-based IVs with human-annotated labels.

III. Empirical Study

Our pre-study analysis aims to identify effective IVs in validating test inputs generated by distinct TIGs. Our findings in this section offer empirical evidence to enhance the existing testing framework. We have studied four automated IVs and three TIGs, all of which are representative in the realm of DL testing and anomaly detection research.

A. Empirical Setup

The pipeline of our empirical setup is summarized in Figure 1. Initially, we randomly select 100 seed images from the dataset and pass them to each TIG to generate test inputs. The generated test inputs are then assessed by domain experts and IVs for validity identification. Binary validity labels (0 = invalid, 1 = valid) provided by human assessors are regarded as ground truth. With available ground truth, we assess the trustworthiness of automated validators. All experiments were performed on machines equipped with Intel Xeon Silver 4210 CPU, and Nvidia GeForce RTX 3090-Ti GPU.

1) Datasets & Models: We evaluate two datasets, MNIST [35] and Udacity self-driving car challenge dataset [36], each with one DNN model that achieve competitive performance. **MNIST** contains 70000 grayscale images of handwritten digits from 0 to 9, each with dimension 28×28. There are 60000 training images and 10000 test samples. We use a classification model [6] with 5-layer ConvNet with max-pooling and dropout layers, which achieves state-of-the-art performance (99.37% accuracy). **Udacity Driving Dataset** contains 101,396 training and 5614 testing RGB images captured by cameras mounted behind the windshield of a human-driving car. The steering angle applied by the human driver is also recorded along with each image frame. We use the competitive DAVE-2 self-driving car model from Nvidia [37] with dropout layers, achieving 99.1% accuracy[1].

2) Human Evaluation: We analyze the validity of each generated test input from the human perspective. We invite 4 volunteers to complete the questionnaire. We designed the questionnaire with one question for each image generated by each testing technique [15]. For the test inputs that two assessors disagree, we invite a third person to review the case and provide a majority vote. For the MNIST dataset, we

[1]we report 1-MSE (Mean Squared Error) as the accuracy.

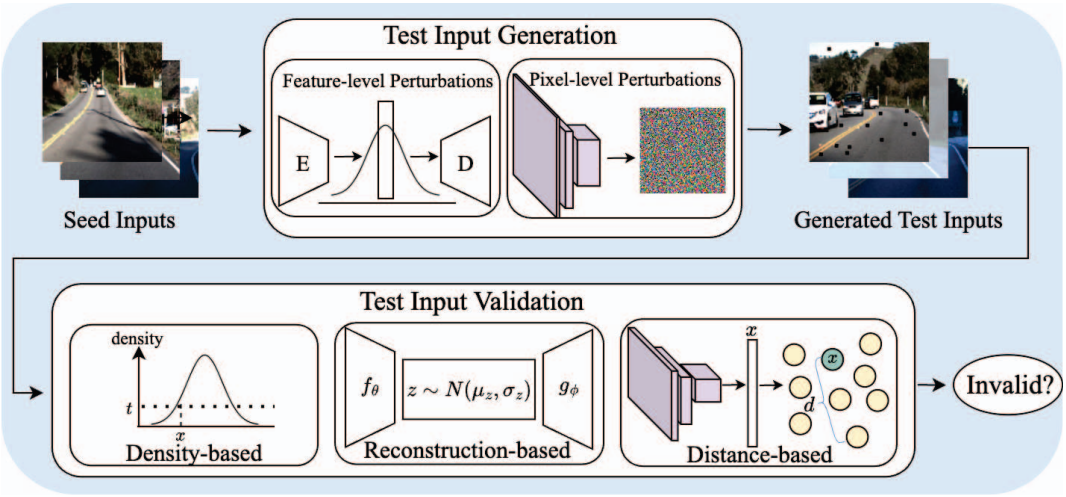


Fig. 1: Workflow of Input Validation: 1) Test Input Generation from seed inputs; 2) Test Input Validation by automated IVs.

ask the assessors to classify the image with eleven choices, including classes 0 to 9, "Not a handwritten digit", and "It is a digit but belongs to other classes, if yes, please specify" (e.g., number 11). The latter two choices indicate the input image is out of the valid input domain and should not be predicted by the model. For the Udacity dataset that captures real-world driving images, we ask whether the generated inputs are likely to be captured in real-world driving scenarios (e.g., can see the road clearly and can confidently steer the car under the given driving condition).

Reproducibility. The implementation and data are available at https://github.com/InputValidityRepo/InputValidity.

B. Studied TIGs

1) Description: Existing TIGs [4]–[7], [11] have been proven to efficiently generate a large number of misbehavior-inducing test inputs. However, they lack assessing the validity of the generated inputs, which cannot guarantee effective testing. In this section, we analyze the quality of generated inputs from three different testing techniques with human evaluation: DeepXplore [5] with three types of modifications to images: occlusion by a small rectangle (DO: occl), lighting effects (DL: light), and occlusion by multiple tiny black rectangles (DB: black), DLFuzz (DF) [11] and Sinvad (SV) [6]. Details for each TIG are summarized in Table I. Please refer to the original papers for further information. To ensure a fair comparison, for each dataset, we use the *same* DNN model under test for all TIGs.

2) Observation: We examine the quality of generated test inputs with ground truth validity labels. The results are shown in the last column of Table II. In summary, for the MNIST dataset, all tested TIGs can produce invalid inputs, in which SV produces the highest number (48%) of invalid inputs. Techniques using pixel-level perturbations (DeepXplore with its variants and DF) tend to produce fewer invalid inputs. For Udacity, all test inputs generated by DO, DB, and DF are valid,

TIGs	Access	Description	Objectives
DeepXplore [5] (DO, DL, DB)	White-box	Pixel-level perturbations	Neuron Coverage Misbehaviors
DLFuzz (DF) [11]	White-box	Pixel-level perturbations	Neuron Coverage Misbehaviors
SINVAD (SV) [6]	Black-box	Feature-level perturbations	Misbehaviors

TABLE I: Summary of studied TIGs

TABLE II: TIG Effectiveness: Input Validation results (unit: % of valid) with automated and human validators. For each IV, we highlight the best TIG in bold.

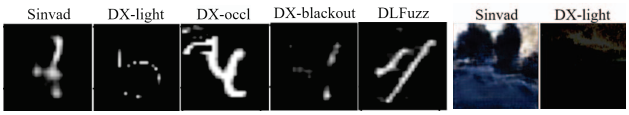
	MNIST					Udacity				
TIG	DAIV	DeepSVDD	DeepKNN	PCNN	Human	DAIV	DeepSVDD	DeepKNN	PCNN	Human
DO	2	32	81	25	96	95	**100**	97	76	**100**
DL	86	95	60	86	**99**	89	80	74	**93**	75
DB	**100**	90	60	**99**	80	79	**100**	**98**	90	**100**
DF	26	91	**85**	0	92	63	**100**	85	0	**100**
SV	**100**	**98**	81	14	52	**100**	**100**	**98**	25	0

while none for SV. We found that the test inputs generated by SV are excessively blurred, as depicted in Figure 2(b). This may attributed to the amplified distortions brought by feature-level perturbation.

Finding 1: All examined TIGs can generate invalid inputs. With human assessment, SINVAD produces the highest number of invalid inputs for both datasets.

C. Studied IVs

1) Description: We explore the effectiveness of PixelCNN++ [20], reconstruction-based DAIV [16], distance-based



(a) MNIST (b) Udacity

Fig. 2: Invalid test inputs for MNIST and Udacity datasets.

DeepKNN [29] and DeepSVDD [31]. Following the established setting in the previous work [15], [16], we train all the IVs on the training set of each dataset, which is regarded as valid data by construction.

In this section, we provide a formulated description of each IV, followed by our experimental observations to the question: *Which IV is Better?* Specifically, with the ground truth labels provided by human assessment, for each IV, we compute True Positive (TP, valid for both automated and human validators), False Positive (FP, valid for automated IV but invalid for humans), True Negatives (TN, invalid for both automated and human validators), False Negatives (FN, invalid for automated IV but valid for humans), and Accuracy ($Acc = \frac{TP+TN}{TP+TN+FP+FN}$).

Notations: For the formulated description in Table III, for PixelCNN++, p denotes the estimated probability density for the test input sample x; i denotes all sub-pixels in x. For DAIV, p denotes the reconstruction probability for x. For DeepKNN, z^k_{train} is the normalized feature vector of the test input's k-th nearest neighbor in the training data; z is the normalized feature vector for x; we set $k = 50$ for MNIST and $k = 100$ for Udacity. For DeepSVDD, c is the hypersphere center, ϕ is the feature encoder.

2) Observation: We analyze the accuracy of each IV by comparing it with manual labels. **IV Effectiveness and recommendations:** Table IV provides the results of TP, FP, TN, FN, and Accuracy for measuring whether automated IVs reach an agreement with humans. For MNIST, on average of all generated test cases, DeepKNN achieves the highest accuracy (72%). Similarly, distance-based DeepSVDD takes the second spot. For Udacity, the average accuracy indicates that feature-based IVs (DeepSVDD and DeepKNN) outperform DAIV and PixelCNN++. Specifically, DeepSVDD reaches the highest accuracy under 4 out of 5 cases with an average of 77%. Intuitively, for image data with higher complexity, it is more difficult to directly model the distribution on pixel space (PixelCNN++) or reconstruct the image (DAIV). Another interesting phenomenon is that all IVs except for PixelCNN++ perform poorly on Sinvad, with only 0% to 1% accuracies.

Finding 2: The average accuracy of IVs ranges from 49% to 77%. Generally, we suggest using feature-based IVs for datasets with higher complexity. On average, DeepKNN and DeepSVDD achieve the best performance for MNIST and Udacity, respectively.

Algorithm 1 Enhanced Testing Framework by Joint Opt.

Require: X - Seed inputs; obj_1 - Original objectives of TIG; obj_2 - Distribution-aware objectives of IV; DNN - DNN under test; hyperparameters - $\{lr, \gamma, t, iters\}$

1: $test\_suite = \{\}$
2: **for** x **in** X **do**
3: **for** i=1 **to** iters **do**
4: $obj = obj1 + \gamma \times obj2$
5: $G = \partial obj / \partial x$
6: $G = Process(G)$
7: $x = x + lr \times G$
8: **if** Misbehavior(DNN,x) and PassValidityCheck(x,t) **then**
9: $test\_suite.append(x)$
10: break
11: **end if**
12: **end for**
13: **end for**

IV. Our Approach and Results

Based on our findings, we enhance existing white-box testing techniques to generate valid inputs for DNN testing by incorporating distribution awareness, which prevents the generated input from deviating significantly from the learned distribution. We examine the quality of generated test cases by both humans and automated IVs. With **the same** 100 seed inputs, the results show that our enhanced framework can generate up to 10% more valid test inputs.

A. Enhanced Testing Framework

To introduce distribution awareness, we perform joint optimization, in which the best IV's objective is optimized together with each TIG's objective (e.g., DAIV for DeepXplore-blackout for MNIST, as shown in Table IV). It should be emphasized that this improvement is specifically applicable to white-box testing techniques that employ gradient-based optimization, and it does not extend to black-box techniques.

Algorithm 1 elaborates details of the enhanced testing framework achieved through joint optimization. In the **Require** lines. For X, we use the same 100 seed inputs as described in Section III-A. For obj_2, we use probability density $\log p(x)$ for PixelCNN++, reconstruction probability $p(x)$ for DAIV, negative feature distance $-d_k(x)$ between x and training data for DeepKNN, and negative feature distance $-d_c(x)$ between x and the hypersphere center c for DeepSVDD. For hyperparameters, we set $\gamma = 0.01$, and determine t as described in Table III. lr and $iters$ values are set as in the original TIG works. We begin by initializing the *test_suite* to store the generated test inputs (Line 1). The generation process starts with looping over each seed image x in X (Line 2-3). We construct a joint objective obj by the weighted sum of $obj1$ and $obj2$ (Line 4). We maximize the joint objective and generate a new test input by performing gradient ascent, note that the gradients G are processed by domain constraints, if any. (Line 5-7). We add the generated input to the test suite if it is confirmed valid by the selected IV (ValidityCheck) and causes model misbehavior (Line 8-10).

IVs (type)	Description	Formulae (Notations)	Validity	Threshold t
PixelCNN++ [20] (Density-based)	Statistically models probability distribution for valid data	$\log p = \sum_i \log p(x_i \| x_{<i})$	valid if $\log p \geq t$, otherwise invalid	Set t at where 95% valid test data are correctly classified
DAIV [16] (Reconstruction-based)	Trains a probabilistic detector that maximizes the likelihood of valid data	$p = \frac{1}{L}\sum_{l=1}^{L} p_\theta(x \| \mu_{z_l}, \sigma_{z_l})$	valid if $p \geq t$, otherwise invalid	Use a self-defined invalid dataset to pick t with highest F-measure
DeepKNN [29] (Distance-based)	Compute k-NN distance on feature space	$d_k = \|z - z^k_{train}\|_2$	valid if $d_k \leq t$, otherwise invalid	Set t at where 95% valid test data are correctly classified.
DeepSVDD [31] (Distance-based)	Encode valid data representations into a DNN-based hypersphere	$d_c = \|\phi(x) - c\|_2$	valid if $d_c \leq t$, otherwise invalid	Set t as the radius of the trained hypersphere.

TABLE III: Description of studied IVs

TABLE IV: IV Effectiveness: IV vs. human labelling. For each TIG, we highlight the best IV's Acc (%) in bold.

IV	TIG	MNIST					Udacity				
		TP	FP	TN	FN	Acc	TP	FP	TN	FN	Acc
DAIV	DO	2	0	4	94	6	95	0	0	5	95
	DL	85	1	0	14	85	64	25	0	11	64
	DB	80	20	0	0	**80**	79	0	0	21	79
	DF	24	2	6	68	30	63	0	0	37	63
	SV	52	48	0	0	52	0	100	0	0	0
	Avg	49	14	2	35	51	60	25	0	15	60
DeepSVDD	DO	30	2	2	66	32	100	0	0	0	**100**
	DL	95	0	1	4	**96**	69	11	14	6	**83**
	DB	74	16	4	6	78	100	0	0	0	**100**
	DF	83	8	0	9	83	100	0	0	0	**100**
	SV	52	46	2	0	54	0	100	0	0	0
	Avg	67	14	2	17	69	74	22	3	1	77
DeepKNN	DO	80	1	3	16	**83**	97	0	0	3	97
	DL	60	0	1	39	61	65	9	16	10	81
	DB	54	6	14	26	68	98	0	0	2	98
	DF	82	3	5	10	**87**	85	0	0	15	85
	SV	48	33	15	4	**63**	0	99	1	0	1
	Avg	65	9	8	19	72	69	22	3	6	72
PixelCNN++	DO	24	1	3	72	27	76	0	0	24	76
	DL	85	1	0	14	85	68	25	0	7	68
	DB	79	20	0	1	79	90	0	0	10	90
	DF	0	0	8	92	8	0	0	0	100	0
	SV	6	8	40	46	46	0	25	75	0	**75**
	Avg	39	6	10	45	49	47	10	15	28	62

TABLE V: Effectiveness of Joint Optimization (% valid). The positive/negative/equal signs indicate the improved techniques produce more/fewer/the same number of valid test inputs compared to the original techniques with the same seed inputs.

Data	TIG	DAIV	DeepSVDD	DeepKNN	PCNN	Human
MNIST	DO+KNN	3(+)	30(-)	100(+)	30(+)	98(+)
	DL+SVDD	83(-)	100(+)	80(+)	83(-)	99(=)
	DB+DAIV	100(=)	96(+)	62(+)	100(+)	83(+)
	DF+KNN	44(+)	98(+)	100(+)	5(+)	95(+)
Udacity	DO+SVDD	93(-)	100(=)	96(-)	78(+)	100(=)
	DL+SVDD	84(-)	100(+)	85(+)	93(=)	85(+)
	DB+SVDD	82(+)	100(=)	94(-)	91(+)	100(=)
	DF+SVDD	80(+)	100(=)	100(+)	0(=)	100(=)

B. Results

Validity Assessments: The validity assessment results are presented in Table V. Note that all test inputs selected with the chosen IV are 100% valid due to the ValidityCheck of our method. Comparing these results to Table II, we observe that 78% of the time, automated IVs agree that joint optimization is beneficial in generating more or the same number of valid test inputs. It is also crucial to consider IV's reliability when they do not acknowledge the improvement. Human assessors consistently provide positive feedback (100% of the time), in which we observe a 2% to 10% increase in the number of valid test inputs.

Practicability: In our method, for each TIG, we select the best IV for joint optimization based on our pre-study results on 100 seed inputs. To apply the enhanced framework in practice, we suggest two ways, the first would be to reduce the number of seed inputs for pre-study (i.e., selecting only 10 samples can obtain an initial idea). The second would be directly referring *Finding 2*, which suggests applying feature-based IVs for both datasets. We plan to experimentally validate this assumption on more datasets in our future work.

Result: Domain experts have consistently (100% of the time) confirmed the effectiveness of our enhanced framework in valid input generation, demonstrating an up to 10% increase in the number of valid inputs.

V. Conclusion

In this paper, we undertake an empirical study and enhance the testing framework by incorporating distribution awareness. Our investigation reveals that the current TIGs can generate invalid inputs, and the average accuracy of automated IVs ranges from 49% to 77%.

Based on these findings, we improve the current testing framework by integrating IVs' distribution-aware objectives to generate more valid inputs. The results demonstrate a 2% to 10% increase in the number of valid inputs, as assessed by human evaluation. In future work, we aim to extend our approach to other fields to enhance the completeness and understand the generalizability of our findings.

Acknowledgement

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (6000796, 9229109, 9229098, 9220103, 9229029).

References

[1] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3642–3649.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," *Advances in neural information processing systems*, vol. 25, 2012.

[4] H. Zhou, W. Li, Z. Kong, J. Guo, Y. Zhang, B. Yu, L. Zhang, and C. Liu, "Deepbillboard: Systematic physical-world testing of autonomous driving systems," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 347–358.

[5] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[6] S. Kang, R. Feldt, and S. Yoo, "Sinvad: Search-based image space navigation for dnn image classifier test input generation," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 521–528.

[7] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "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)*. IEEE, 2018, pp. 132–142.

[8] Z. Kong, J. Guo, A. Li, and C. Liu, "Physgan: Generating physical-world-resilient adversarial examples for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 254–14 263.

[9] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[10] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Deepconcolic: Testing and debugging deep neural networks," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 111–114.

[11] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "Dlfuzz: Differential fuzzing testing of deep learning systems," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 739–743.

[12] J. H. Hayes and J. Offutt, "Input validation analysis and testing," *Empirical Software Engineering*, vol. 11, pp. 493–522, 2006.

[13] H. Liu and H. B. K. Tan, "Covering code behavior on input validation in functional testing," *Information and Software Technology*, vol. 51, no. 2, pp. 546–553, 2009.

[14] S. Hanna and M. Munro, "Test case generation for semantic-based user input validation of web applications," *International Journal of Web Engineering and Technology*, vol. 13, no. 3, pp. 225–254, 2018.

[15] V. Riccio and P. Tonella, "When and why test generators for deep learning produce invalid inputs: an empirical study," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1161–1173.

[16] S. Dola, M. B. Dwyer, and M. L. Soffa, "Distribution-aware testing of neural networks using generative models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 226–237.

[17] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 359–371.

[18] J. Yang, R. Xu, Z. Qi, and Y. Shi, "Visual anomaly detection for images: A survey," *arXiv preprint arXiv:2109.13157*, 2021.

[19] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey. arxiv," *arXiv preprint arXiv:2110.11334*, 2021.

[20] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[21] W. Hu, J. Gao, B. Li, O. Wu, J. Du, and S. Maybank, "Anomaly detection using local kernel density estimation and context-based regression," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 2, pp. 218–233, 2018.

[22] R. Hinami, T. Mei, and S. Satoh, "Joint detection and recounting of abnormal events by learning deep generic knowledge," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3619–3627.

[23] L. Zhang, J. Lin, and R. Karim, "Adaptive kernel density-based anomaly detection for nonlinear systems," *Knowledge-Based Systems*, vol. 139, pp. 50–63, 2018.

[24] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[25] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, and N. Ding, "Gan-based anomaly detection: A review," *Neurocomputing*, vol. 493, pp. 497–535, 2022.

[26] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient gan-based anomaly detection," *arXiv preprint arXiv:1802.06222*, 2018.

[27] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. Springer, 2019, pp. 622–637.

[28] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6536–6545.

[29] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," in *International Conference on Machine Learning*. PMLR, 2022, pp. 20 827–20 840.

[30] T. Dietterich, "A study of distance-based machine learning algorithms," Ph.D. dissertation, Ph. D. Thesis, computer Science Dept., Oregon State University, 1995.

[31] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[32] I. Dunn, H. Pouget, D. Kroening, and T. Melham, "Exposing previously undetectable faults in deep neural networks," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 56–66.

[33] I. Dunn, L. Hanu, H. Pouget, D. Kroening, and T. Melham, "Evaluating robustness to context-sensitive feature perturbations of different granularities," *arXiv preprint arXiv:2001.11055*, 2020.

[34] J. Kauffmann, L. Ruff, G. Montavon, and K.-R. Müller, "The clever hans effect in anomaly detection," *arXiv preprint arXiv:2006.10609*, 2020.

[35] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[36] Udacity, "Using deep learning to predict steering angles." 2016. [Online]. Available: https://medium.com/udacity/challenge-2-using-deep-learning-to-predict-steering-angles-f42004a36ff3

[37] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.