2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)

Structural Coverage Criteria for Neural Networks Could Be Misleading

Zenan Li, Xiaoxing Ma, Chang Xu and Chun Cao
State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing University, Nanjing, China
lizenan1995@foxmail.com, {xxm, changxu, caochun}@nju.edu.cn

***Abstract*—There is a dramatically increasing interest in the quality assurance for DNN-based systems in the software engineering community. An emerging hot topic in this direction is structural coverage criteria for testing neural networks, which are inspired by coverage metrics used in conventional software testing. In this short paper, we argue that these criteria could be misleading because of the fundamental differences between neural networks and human written programs. Our preliminary exploration shows that (1) adversarial examples are pervasively distributed in the finely divided space defined by such coverage criteria, while available natural samples are very sparse, and as a consequence, (2) previously reported fault-detection "capabilities" conjectured from high coverage testing are more likely due to the adversary-oriented search but not the real "high" coverage.**

***Keywords*-Software Testing, Neural Networks, Coverage**

I. Introduction

Deep Neural Networks (DNNs) have demonstrated amazing performance in various tasks such as image classification and speech recognition [1]. They are also increasingly adopted in safety-critical application scenarios such as medical diagnostics [2] and self-driven cars [3]. Therefore how to assess and improve the reliability of DNN-based systems becomes a highly relevant problem and has attracted a lot of research [4].

A particular issue is the testing of trained neural network models. In this paper we focus on DNN-based classifiers. Different from the testing phase included in the training process that gauges the generalization of a model, here a neural network is treated as a piece of software and intentionally exercised to find potential defects when used in the real world. Depending on the application scenario, the defects hunted for can be

- *Natural inputs* that will be misclassified by the neural network. Natural inputs are those appearing in the real world, and assumably distribute similarly as the training data of the network.
- *Adversarial inputs* that can fool the network. An adversarial input, or adversarial example [5], is fabricated by, e.g., adding a well-designed perturbation to a genuine example [6].

For applications used in a friendly environment, one only needs to consider misclassified natural inputs. However in a hostile environment the threats of adversarial inputs must be taken into account. As discussed later, the distinction between natural and adversarial inputs is important.

It is challenging to test a neural network sufficiently. Different from human written programs with unambiguous intended behavior on any legitimate input, neural networks are usually trained to provide only statistical guarantees such as accuracy and loss under the intangible I.I.D. assumption [7]. In addition, the logical interpretation of DNNs' behavior on individual examples is still an open problem [5].

Recently, inspired by the white-box testing of conventional software, a variety of structural coverage criteria has been proposed to gauge the defect-finding capability (or fault-detection capability) of DNN testing [8]. In addition to measuring the sufficiency of testing, they are also intended to guide the automated generation of test inputs and to improve DNN performance [9]. These researches are very inspiring, and some of them have been recognized with best paper awards on major academic conferences [9], [10].

However, our preliminary exploration shows that these proposed structural coverage criteria could be misleading if used without understanding their underlying principles and application contexts. This is because

1) The distribution of defects in human-written programs is fundamentally different from that in DNNs. As shown later, adversarial examples are pervasively distributed over the finely divided space defined by given coverage criteria. On the other hand, the distribution of available natural inputs are very sparse, not to mention the rare misclassified natural inputs. That is to say, these structural coverage criteria could be too coarse for adversarial inputs and at the same time too fine for misclassified natural inputs.
2) Previously reported fault-detection "capabilities" of high coverage testing are more likely due to the adversary-oriented search but not the structural coverage. Our exploration also shows that the number of adversarial examples found by coverage-oriented input generation can be easily manipulated.
3) Our initial experiments with natural inputs denied the correlation between the number of misclassified inputs in a test set and its structural coverage on the associated neural networks.

In the rest of this paper, we first briefly introduce structural coverage criteria for DNNs recently proposed by different authors. After that we present our analyses and experiments supporting the above arguments.

978-1-7281-1758-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-NIER.2019.00031

II. Structural coverage for DNN Testing

A Deep Neural Network (DNN) is a network with multiple layers of artificial neurons between the input and output layers [1]. It encodes the mathematical mapping from inputs to outputs, which approximates the distribution hidden in the training data, with a cascading composition of simple functions implemented by the neurons. Due to its approximative nature, the possible biased training data, and the over-fitting or under-fitting in the training process, a DNN can misbehave on some inputs. Thus it needs to be sufficiently tested before being put into use.

In traditional white-box software testing, the structural information of the program under testing is leveraged to ensure that the program has been thoroughly tested. Various coverage criteria have been proposed for this purpose [11] Inspired by these criteria, recently some authors proposed several structural coverage criteria for the testing of neural networks [9], [10], [12], [13]. Due to the space limit, we can only introduce them briefly:

- *Neuron Coverage*: Neuron coverage of a test set is measured by the percentage of neurons that are activated by the inputs in the set [9].
- *Neuron-Output coverages*: The neuron coverage is so coarsely-grained that it is very easy to achieve a nearly 100% coverage rate. It can be refined with the consideration of the output value of each neuron. For example, DeepGauge uses *k*-multisection Neuron Coverage that divides the output range of each neuron into *k* chunks [10].
- *Combinatorial Coverages*: Finer-grained criteria can also be derived by combining the activation states of multiple neurons. For example, inspired by combinatorial testing, DeepCT introduces *t*-way combination sparse coverage and *t*-way combination dense coverage [13], and inspired by the MC/DC coverage [14], DeepCover introduces several coverage criteria including SS-Cover [12].

III. Testing for adversarial inputs

All of the coverage criteria discussed above have been used to direct the generation of adversarial inputs. This practice is further used as the proof of the effectiveness of the criteria. However, we are skeptical about this logic. Practically, there exist more effective and efficient techniques for generating adversarial examples [15], [16]. More importantly, we suspect that these structure criteria do not provide any more information about the actual quality of the models than those already leveraged by existing adversary-oriented searches.

The intuitions behind the usefulness of structural coverage for human-written programs are (1) the *homogeneity* of inputs covering the same part of a program – either all or none of them induces an error, so testing efficiency can be improved by avoiding duplications; and (2) the *diversity* of inputs indicated by coverage metrics, so testing efficacy can be achieved by enforcing high coverage to touch corner cases in which rare errors may hide.

However, DNNs behave differently. The homogeneity is broken by the fact that adversarial inputs to DNNs are pervasively distributed over the input space, together with correct classified inputs. The diversity is also in question when the coverage-oriented search is compared to the adversary-oriented search.

A. Pervasiveness of adversarial inputs

Let us consider an extremely fine-grained criterion that is defined by the combination of the activation state of each neuron, assuming ReLU as the activation function. This criterion divides the input space of a network with n neurons into 2^n polyhedra. In the area of a polyhedron, the neural network degenerates into a linear classifier.

Table I
The pervasiveness of adversarial inputs

	hidden layers	**adversarial**	**random**
N_1	67x22x63	99.87%	100%
N_2	59x94x56x45	86.63%	100%
N_3	72x61x70x77	82.21%	99.98%
N_4	65x99x87x23x31	76.02%	96.80%
N_5	49x61x90x21x48	72.48%	94.62%
N_6	97x83x32	96.43%	99.87%
N_7	33x95x67x43x76	82.12%	94.00%
N_8	78x62x73x47	90.87%	98.77%
N_9	87x33x62	97.54%	100%
N_{10}	76x55x74x98x75	66.37%	98.81%

Table I presents the percentages of polyhedra that contain adversarial examples (column '**adversarial**'). N_1 to N_{10} are the 10 MNIST-trained neural networks used in the evaluation of DeepCover [12]. In the experiment we randomly took 10,000 inputs from the test set, and for each of them we searched in its corresponding polyhedron for an adversarial example. The percentages are even higher if we are not limited to the test set but take inputs freely for each randomly selected polyhedron (column '**random**').

The high percentages in Table I unambiguously indicate that adversarial examples are pervasively distributed in the space finely divided by coverage criteria such as MC/DC and DeepCT. Similar result are also observed for Neuron-Output coverages. This is also consistent with the recent research about properties of adversarial examples [17]. This observation intuitively contradicts the effectiveness of the structural coverage criteria for neural networks.

B. Similarity between adversarial and coverage perturbations

All the structural criteria discussed above have been reported to be effective in directing the generation of adversarial inputs. However, there is no evidence that with

the extra information provided by the structural coverage, one can achieve better efficiency and efficacy than existing adversarial example search methods such as C&W [16] and FGSM [15].

We tend to explain the effect of high coverage in finding adversarial samples as the similarity between adversarial and coverage perturbations. Here adversarial perturbation is computed in the adversary-oriented search, while coverage perturbation is added to an original input to generate a new input for covering a neighbor polyhedron. Both adversarial perturbation and coverage perturbation make use of the structural information of a trained neural network.

To be precise, for a deep neural network with the ReLU activation function, given an input $\boldsymbol{x}$, the linear behavior of the neural network with n layers in the polyhedron to which the input $\boldsymbol{x}$ belongs can be written as: $f(\boldsymbol{x}) = \boldsymbol{w}_n^T(\boldsymbol{w}_{n-1}^T(\ldots(\boldsymbol{w}_1^T\boldsymbol{x}+b))+\boldsymbol{b}_{n-1})+\boldsymbol{b}_n$, where $f(\boldsymbol{x})$ is the logit, i.e., the input of the softmax layer. Thus adversarial perturbations can fool the deep neural network in the direction of gradient $\nabla f(\boldsymbol{x})$. If the target label of an adversarial attack is l, the gradient of adversarial perturbation is $\nabla f(\boldsymbol{x}) = \boldsymbol{w}_{(n,l)} \cdot \boldsymbol{w}_{n-1} \cdots \boldsymbol{w}_1$, where $\boldsymbol{w}_{(n,l)}$ is the weight vector of the l-th neuron at the last layer.

To improve coverage, one needs a perturbation that changes the output or the activation state of neuron $n_{(k,i)}$. It is the gradient of the neuron output $\nabla h(\boldsymbol{x}) = \boldsymbol{w}_{(k,i)} \cdot \boldsymbol{w}_{k-1} \cdots \boldsymbol{w}_1$, where $\boldsymbol{w}_{(k,i)}$ is the weight vector of the target neuron.

It follows from the two equations of $\nabla f(\boldsymbol{x})$ and $\nabla h(\boldsymbol{x})$ that the deeper the layer of the changed neuron is, the smaller the difference between coverage perturbation and adversarial perturbation would be.

We have also measured the similarity empirically. We are forced to omit the details by the space limit, but the result confirmed the above analysis.

On one hand, this similarity explains why the structural coverage is effective in the search for adversarial inputs. On the other hand, it could also implies that the structural coverage would not add too much to the adversary-oriented search.

C. Manipulatable robustness measurement

Another proposed usage of structural coverage criteria is to measure the robustness of neural networks [12]. That is, by generating a high-coverage test input set for a neural network, the less adversarial inputs found in the set, the more robust the neural network is believed to be. However, as discussed earlier, adversarial examples are pervasive, and thus the number of adversarial examples found in the course of achieving high coverage highly depends on the search method used. In the following, we show that this measurement can be easily manipulated.

Take the SS-Cover coverage as an example. Table II shows the percentages of adversarial examples found with the method used in DeepCover. However, with a different search method, we can get a completely different result. Column '**max**' shows the percentages of adversarial examples with a technique to maximize the values. This method finds adversarial examples in each same polyhedron as the one generated by DeepCover for SS-Cover coverage. So the coverage remains the same.

We also compute the average l_1 norms of the adversarial samples found by the maximizing technique (column 'l_1 **max**') and compare them with those of DeepCover (column 'l_1 **DpCv**'). The result confirms that the adversarial examples found by the two methods are similar in terms of their degrees of perturbation.

Table II
ADVERSARIAL EXAMPLE PERCENTAGES UNDER HIGH COVERAGE DATA

	coverage	**DeepCover**	**max**	l_1 **DpCv**	l_1 **max**
N_1	99.9%	17.4%	92.0%	56.6	54.0
N_2	99.7%	7.2%	93.5%	49.3	52.7
N_3	99.5%	7.3%	94.3%	54.2	53.6
N_4	98.9%	7.4%	91.4%	54.0	60.0
N_5	94.1%	8.8%	87.6%	45.5	46.5
N_6	100%	7.6%	95.1%	52.9	54.3
N_7	90.9%	6.2%	95.6%	44.0	45.9
N_8	99.9%	8.1%	88.5%	56.5	54.0
N_9	99.9%	12.0%	91.1%	57.4	56.7
N_{10}	86.7%	5.8%	83.3%	52.9	52.8

Table II suggests that the percentages of adversarial examples can be manipulated to be much higher than those reported by DeepCover. In addition, the correlation coefficient between the two columns are very small (0.0618). The same problem also occurred to DeepCT. This result indicates that the usage of structural coverage criteria in the measurement of neural network robustness can be questionable if its context is overlooked.

IV. TESTING FOR MISCLASSIFIED NATURAL INPUTS

As aforementioned, adversarial examples are pervasive for DNNs. However, up to date there is no practical and effective defense method for adversarial attacks [18]. Fortunately, if a DNN is used in a friendly application scenario, one only needs to consider natural inputs that may appear in the real world. In this context, a good coverage criterion should be able to characterize the fault-detection capability of a given natural input set.

However, all our studied existing coverage criteria have only been evaluated with artificial inputs generated toward high coverage. As explained above, their capabilities of finding adversarial samples might be due to the adversary-oriented search, and does not necessarily correspond to the real capability of detecting misclassification of natural inputs.

A direct evaluation of the effectiveness of these coverage criteria is difficult, because it is non-trivial to fairly generate same-size natural input sets with significantly different coverages. However, if the coverage criteria for neuron

networks work similarly as those for conventional software, a randomized test set with more error-inducing inputs will touch more unnoticed corner cases, and thus shall have a higher coverage. So we conducted an experiment with natural input sets with different levels of error rate, to see how the coverage varies with the error rate.

Table III presents the result of the experiment with the k-multisection criterion of DeepGauge [10]. The experiment used four variants of LeNet-5 networks for the MNIST dataset [19] and two pre-trained networks (VGG-19 and ResNet-50) for the ImageNet dataset [20]. The size of the natural test input sets was fixed to 5,000.

Table III
COVERAGE DATA WITH NATURAL INPUTS OF DIFFERENT ERROR RATES

model	k-value	err.=0.01%	err.=1.0%	err.=5.0%	err.=10.0%
LeNet-5	$k = 1,000$	65.3%	65.3%	-	-
	$k = 10,000$	20.3%	20.3%	-	-
LeNet-5 mutation1*	$k = 1,000$	64.1%	64.2%	64.4%	64.5%
	$k = 10,000$	19.7%	19.7%	19.8%	19.8%
LeNet-5 mutation2*	$k = 1,000$	67.3%	67.4%	67.8%	68.3%
	$k = 10,000$	20.7%	20.8%	20.8%	20.9%
LeNet-5 mutation3*	$k = 1,000$	65.2%	65.2%	65.2%	65.1%
	$k = 10,000$	19.1%	19.1%	19.1%	19.1%
VGG-19	$k = 1,000$	37.2%	37.2%	37.2%	37.0%
	$k = 10,000$	14.7%	14.7%	14.7%	14.6%
ResNet-50	$k = 1,000$	51.1%	51.1%	51.0%	50.4%
	$k = 10,000$	20.7%	20.7%	20.7%	20.7%

*Three mutated models are trained by changing the labels of training data from 0 to 6, 2 to 3, 4 to 9, respectively.

No significant change in the coverage was observed when error rates grew from 0.01% to 10%. Similar results were also observed for the neuron coverage criterion of DeepXplore [9]. These results suggest that such criteria might be ineffective for fault detection of DNNs with natural inputs. We have not conducted experiments with with the SS-Cover and DeepCT's criteria, because they are too fine-grained to let available natural inputs achieve a meaningful coverage.

V. CONCLUSION

Though preliminary, our experiments and analyses suggest that previously proposed structural coverage criteria for neural networks could be misleading. In our opinion, a useful criterion, if used in a friendly context, should be able to distinguish same-size test sets of natural examples with different levels of misclassification rates. If used in an attack defense context, it should make the generation of adversarial examples more effective or efficient than the adversary-oriented only search. Such a criterion must be based on a deeper understanding to the relationship between the network structure and the distributions of adversarial examples and misclassified natural inputs.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61690204) and the 973 Program of China (Grant No. 2015CB352202). Correspondence should be addressed to Xiaoxing Ma.

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.
[2] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843–2851.
[3] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.
[4] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "cleverhans v1.0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.
[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[6] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," *arXiv preprint arXiv:1710.11342*, 2017.
[7] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.
[8] A. Odena and I. Goodfellow, "TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing," *ArXiv e-prints*, Jul. 2018.
[9] 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*. ACM, 2017, pp. 1–18.
[10] L. Ma, F. Juefei-Xu, F. Zhang, et al., "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 120–131.
[11] H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 366–427, Dec. 1997.
[12] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.
[13] L. Ma, F. Zhang, M. Xue, B. Li, Y. Liu, J. Zhao, and Y. Wang, "Combinatorial Testing for Deep Learning Systems," *ArXiv e-prints*, Jun. 2018.
[14] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, and L. K. Rierson, "A practical tutorial on modified condition/decision coverage," 2001.
[15] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples. corr (2015)."
[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.
[17] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.
[18] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," *arXiv preprint arXiv:1802.08686*, 2018.
[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[20] 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*, vol. 115, no. 3, pp. 211–252, 2015.