2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Structure-Based Testing Criteria and Testing Case Generation for Deep Learning Systems

Yining Chen, Jianghua Lv, Fengming Dong and Hexuan Li

Beihang University, Beijing 100191, China
serchen@buaa.edu.cn

Abstract. Deep neural networks (DNN) are currently the basis of many modern AI ap-plications has been widely applied in various domains. As more safety-oriented fields (autonomous driving, medical diagnosis, etc.) begin to use DNN, people have put forward new requirements for DNN. Not only the accuracy of DNN and other objective indicators be excellent, but also have robustness and ability to handle various corner cases. It is important to test the adequacy of the deep neural network model, design appropriate evaluation indicators, build a complete test evaluation system. However, deep neural network computing like a black box, a slight disturbance to the input may cause errors in the final output of the model. Therefore, it is important to test the adequacy of the deep neural network model, design appropriate evaluation indicators, build a complete test evaluation system. We prove that there are differences in the internal structure of neural networks for different types of input. Based on this discovery, we proposed Multi-Layer test criteria based on the neural network structure. To quantify and analyze the changes in the internal structure of neural networks under different types of input, this paper pro poses an algorithm for mapping the deep neural network to tree structure data. Finally, a Multi-Layer test criteria based on the neural network structure is proposed to guide the generation of test cases, which can generate high-quality test cases.

Keywords: Deep neural network, White box testing, test case generation, test effectiveness

1 Introduction

Deep learning[1,2] is part of machine learning. It belongs to artificial neural networks based on data representation learning. Data can usually be represented in a variety of ways. In the field of machine learning[3], developers need to manually extract effective feature expressions from original data, which is a very complicated and skillful task. But in the field of deep learning, users only need to process the data through unsupervised or semi- supervised[4] feature learning methods[5] and leave the rest to the deep neural network.

There are many excellent deep learning architectures[5,6], such as convolutional neural networks, recurrent neural networks and deep belief networks. Deep learning experiences significant progress over the past decades in achieving[7,8,9] competitive

performance of human intelligence in many cutting-edge applications such as image processing , speech recognition [10], autonomous driving[11], medical diagnosis [12] and pharmaceutical discovery[13], which until several years ago were still notoriously difficult to solve programmatically. Most of the model structures only use gradient descent to approach the optimal decision boundary and has no clear definition about convergence problem, so deep learning is often regarded as an unexplainable model.

Therefore, it is very important to properly test the deep neural network model [14], design appropriate evaluation indicators, build a complete test evaluation system, and then make appropriate adjustments and improvements to the model. The current testing of deep neural networks has the following problems: 1) The neuron coverage index design is based on the output of a single neuron as the statistical standard, and does not consider the relationship between multiple neurons and the distribution of multiple layers of neurons. 2) The previous use case generation method did not consider the relationship between the different levels of the neural network. The same use case generation method was adopted for the DNN of different structures, and there was no difference; 3) The lack of DNN under different use cases. The measurement method of the internal changes of the model (the previous DNN testing method did not quantitatively analyze the in-ternal behavior of the DNN) [15].

In response to the above problems: we proved that for different types of input, there are differences in the internal structure of the neural network. Based on this discovery, we proposed a Multi-Layer test criteria based on the neural network structure.

Our contribution mainly lies in:

We propose a Multi-Layer test case generation method based on neural net-work structure. This test criteria verifies the coverage rate of the test case set for the model state from the structural level, so as to verify the adequacy of the test.

We propose a Multi-Layer test case generation method based on neural net-work structure. By transforming the problem of generating target test cases into a joint optimization problem, the purpose of maximizing the difference of the predicted output of the target neural network and maximizing the coverage index of Multi-Layer neurons is achieved.

We have implemented the Multi-Layer deep neural network test framework system, provided the multi-model joint framework test mode framework and the single-model test mode framework based on the actual case display system report to verify the effectiveness of the system.

2 Background

2.1 Neuron Coverage

With reference to the idea of code coverage in traditional software testing, Kexin Pei[1] et al. proposed neuron coverage on deep neural networks as a measure of the test sufficiency of the generated test set for the target deep neural network.

The neuron coverage rate thinks that when using the test set to verify the neural network, if the output value of the neuron in the neural network is higher than the set threshold, then the neuron can be regarded as an activated state. The representative logical structure part is considered to have been executed once, and the coverage rate of a set of test cases to the logical structure of the neural net-work system can be measured by how many neurons in the neural network have been activated. If all the neurons have been activated, then it can be considered that all the logic codes in the deep neural network have been run at least once, and the neuron coverage rate can be considered to reach 100%.

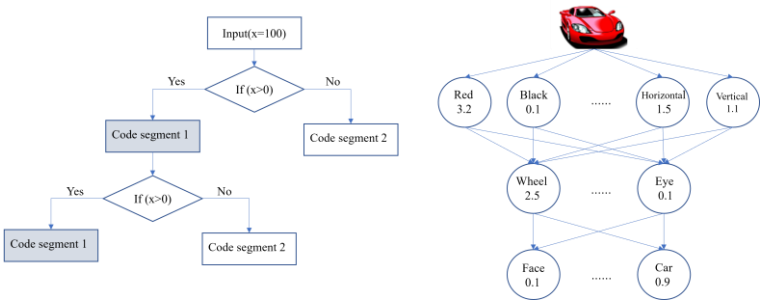

Fig. 1. Comparison of traditional white-box testing and white-box neural network-based testing

Therefore, according to the idea of white box testing [16,17], if you want to fully test the deep neural network, then the test set theoretically needs to execute all the logic branches in the deep neural network at least one side [18], and all the neurons have been activated, which can be considered fully tested.

2.2 Multitree and multitree edit distance

DNN forward propagation and back propagation are both directional, and the number of neurons in the output layer is obviously smaller than the number of nodes in the hidden layer, so we can abstract its internal structure into a tree structure [19].

Inside the DNN, different sample inputs will have a different structure of activated neuron tree structure. In order to quantify the distance between different tree structures: we introduce a multi-tree edit distance Tree edit distance Paaßen [20,21].

The edit distance of a tree is defined as the minimum cost sequence of the node editing operations that transform one tree into another tree between ordered labeled trees. It can be used to measure the similarity of the structure data of the tree. The distance between two trees can be calculated by constructing the following three editing operations on labeled ordered trees:

1. Delete the node and connect its child nodes to the parent node of the dimension order.
2. Insert a node between the existing node and the successive children of the node.
3. Rename the label of the node.

3 Methodology

3.1 Multi-layer Neuron Structure Coverage

Layer Neuron Activation State

In this section, we design a set of DNN test coverage [22] standards from multiple levels, aiming to be able to fully measure the behavioral connections between multiple levels within the DNN network.

We describe the activation state of neurons in a layer of a deep neural network as the number of possible activations of all neurons in this layer is used as the state [23,24], and all possible states are collected. It contains information about the activation state of a single neuron and the neurons in this layer. Through the activation state of the layer of neurons, the distribution of the number of neurons in the layer of activated neurons can be determined, and the upper and lower limits of the number of activated neurons in the layer can also be known.

The definition of Layer Activate State (LSA) is that for a set of test cases $T = x1, x2, \ldots, xn$, the output value out of a certain layer of neuron set is calculated, and t is used as a judgment the threshold of whether a neuron is activated. If $out(n_i, x) > t$ then neuron n is considered to be in an activated state, otherwise it is considered inactive. Use the Enum function to record the number of neurons activated by each test case and remove duplicates. The specific formula is defined as follows:

$$LAS(T, x, N_l) = \{Enum(\sum_{i=1}^{m} n_i) | \forall n_i \in N_l, \forall x \in T, out(n_i, x) > t\} \quad (1)$$

Multi-Layer Neuron Coverage Test Criteria

Starting from the overall neuron structure of the deep neural network, this article constructs three coverage indicators to measure the overall neuron coverage:

Multi-Layer Neuron Coverage (MLNC)

For a set of test case T, calculate the layer neuron activation state (LSA) of this set of test cases for each layer of the deep neural network, and then combine the LSA value of each layer with all the possible activation states of the layer neurons in each layer The ratio of the state is multiplied by the weight of the number of layers, and the final weighted sum is used as the final calculated value of the Multi-Layer neuron coverage. All possible states of the activation state of layer neurons in each layer are defined as: the number of layer nodes plus one.

The specific definition of Multi-Layer neuron coverage test criteria is as follows:

$$MLNC(DNN) = \sum_{l=1}^{n} \frac{LAS(DNN_l)}{n \times Nums(DNN_l)+1} \quad (2)$$

DNN_l refers to a certain layer of the deep neural network model DNN, n is the number of layers of the model DNN, $LAS(DNN_l)$ refers to the activation state of the layer neurons in this layer, and $Nums(DNN_l)$ function refers to the number of neuron nodes

in this layer. Add 1 is to prevent the occurrence of no neuron activation in the current layer.

Test in Train Multi-Layer Neuron Coverage (TTMLNC)
The second test coverage indicator is from the perspective of the training set and the test set. The activation state of the layer neurons covered by the test cases constructed in this paper accounts for the percentage of the activation state of the layer neurons in the training set samples. Define Test in Train Multi-Layer Neuron Coverage test index TTMLNC as:

$$TTMLNC(DNN) = \sum_{l=1}^{n} \frac{LAS_{Test}(DNN_l) \cap LAS_{Train}(DNN_l)}{n \times LAS_{Train}(DNN_l)} \quad (3)$$

$LAS_{Test}(DNN_l)$ represents the number of activation states of neurons in a layer above the test set when the deep neural network model *DNN* is built, and $LAS_{Train}(DNN_l)$ represents the deep neural network model *DNN* is built on the training set. The number of activation states of layer neurons in a layer.

Test in New Multi-Layer Neuron Coverage (TNMLNC)
In order to measure the newly activated state in the test s-et as a percentage of the training set state, the Test in New Multi-Layer Neuron Coverage (TNMLNC) test index is defined as:

$$TNMLNC(DNN) = \sum_{l=1}^{n} \frac{LSA_{Test}(DNN_l) - LSA_{Test}(DNN_l) \cap LSA_{Train}(DNN_l)}{n \times LSA_{Train}(DNN_l)} \quad (4)$$

3.2 DNN Tree Structure distance

We take the highest activation state node in the output layer as the root node of the tree, the neurons activated in the neural network with the same output are similar, and the neurons activated in the internal neural network with different outputs are different. Therefore, we proposed the concept of tree structure set distance.

Through the TNMLNC test criteria, you can measure what percentage of the newly constructed test case set is activated. The state of the previous neural network structure that has not been activated. This is a state that has not been generated from the training set, it can be covered These states that do not appear in the training set but should be covered, so as to build the adequacy of the overall deep neural network white box test process.

For a set of tree structure data, define this set of $t_x \in \sigma_{lebel=x}(x)$ Tree Structure Set Internal Distance (TSSID) as:

$$TSSID(t_x, t_x) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} APTED\left(Tree\left(DNN(t_x)\right), Tree\left(DNN(t_x)\right)\right)}{\left(\frac{n(n-1)}{2}\right)} \quad (5)$$

Where x represents the tree structure data whose neural network output is x and the label of the training sample is x, n represents the number of this set of tree structure data, t_x, $t_x$means that the DNN output and label are both x samples t_x, $APTED$ means

tree edit distance algorithm, DNN represents the neural network being tested, and Tree represents the neural network mapping tree structure algorithm. $\sigma_{lebel=x}(x)$ refers to the tree structure data set in which both the input label and the neural network prediction are x.

For different groups of tree structure data t_x and t_y, they belong to different sample sets, define these two groups of $t_x \in \sigma_{lebel=x}(x)$, $t_y \in \sigma_{lebel=y}(y)$ the outer distance of the tree structure data (Tree Structure Set Outer Distance):

$$TSSOD(t_x, t_y) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} APTED\left(Tree\left(DNN(t_x)\right), Tree\left(DNN(t_y)\right)\right)}{n^2} \quad (6)$$

3.3 Test Case Generation Method

Referring to the idea of software white-box testing, testers need to fully under-stand the running state and logic code of the program, so as to design special test cases to verify the correctness of the software code. Compared with the deep neural network, when designing and selecting new test cases, it is also necessary to fully understand the operating state and internal structure of the deep neural network. Therefore, this paper maps the neural network into a tree structure, and uses the tree structure to measure the performance of the neural network in different inputs. changes below.

In the method of generating test cases for the DNN system through the white box idea, two goals need to be achieved. The first objective is to change the original output of the deep neural network. The second objective is to improve the coverage of Multi-Layer neurons. We set up the two objectives together to con-struct a joint optimization function, and solve the optimal solution of the joint optimization function through gradient descent, and finally obtain the required Multi-Layer test case.

Maximize The Difference in Predicted Output

For objective 1, we guide the final output of multiple models of the same function to be inconsistent, causing errors in the deep neural network, and the output label values of different models are different.

Define $A_p(N_i)$ the output value of the activation function of the i-th neuron in the Prediction layer (the last layer of the neural network) of the deep neural network. By setting a hyperparameter w1 to control the degree of perturbation model output prediction, it can be designed by constructing a loss function that maximizes the difference in the predicted output of the target neural network.

By setting a hyperparameter w_1 to control the degree of perturbation model output prediction, it can be designed by constructing a loss function that maximizes the difference in the predicted output of the target neural network.

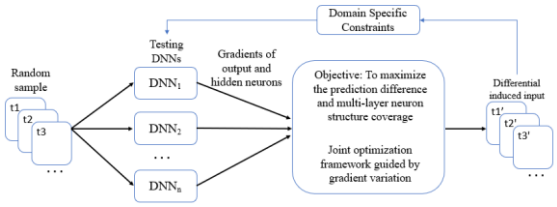

Fig. 2. Multi-layer deep neural network test case generation method

$$obj_1 = \left(\sum_{i=2}^{k} A_p(n_i) - w_i \times A_p\,(N_c)\right) \tag{7}$$

For objective 2, we abstract the neural network into a tree structure and use the designed loss function to push the activated nodes in the neural network to not activated (deactivated), nodes that have not been activated can be activated, resulting in more different structural combinations.

The generated test cases need to be able to activate the tree structure abstract-ed by more different neural networks, because this can cover more neural net-work logic branches, and it also represents a greater coverage of multi-layer neurons.

$$obj_2 = w_2 \times \left(x \sum_{L=1}^{n} \sum_{i=1}^{a_n} A_L(N_i | N_i \in ANS) - y \sum_{L=1}^{n} \sum_{i=1}^{a_n} A_L(N_i | N_i \in UNS)\right) \tag{8}$$

ANS refers to the activated neuron set (Activate Neuron Set), UNS refers to the deactivated neuron set (Deactivate Neuron Set). $A_L(N_i)$ as the ith neuron in the L layer of the neural network.

3.4 Joint Optimization

In this paper, the overall function of the joint optimization problem is constructed, and the two objectives obj_1 and obj_2 are optimized simultaneously by the gradient descent method.

From loss function w_1 can measure the importance of the difference between the sum of the maximum confidence score of the original output of the deep neural network and the confidence scores of other output labels, and w_2 can measure the importance of the internal activated and inactive neuron output change of the neural network after the input sample seed. Two optimizations can be customized by giving different proportions of parameters to w_1 and w_2 the importance of the problem. Therefore, the joint optimization function is fully defined as:

$$obj_{joint} = w_1 \times obj_1 + w_2 \times obj_2 \tag{9}$$

obj_{joint} function can be differential operated in design. Therefore, this paper uses gradient descent algorithm to solve the minimum value of obj_{joint} function. The gradient information obtained each time is transformed to obtain the generated test sample, which interferes with the original input for many times until the generated test sample meets the requirements.

4 Experiment

4.1 Comparison of tree structure distance in neural network

In order to verify whether there are structural differences in the tree structure data mapped by activated neurons after different categories of samples are input into the model. For the input of ten categories from 0 to 9 on the MNIST dataset, we use three different models of LeNet-1, LeNet-4, and LeNet-5 to count the internal distance and external distance of the internal activation node tree structure of the neural network. directly calculate the specific value of the two distances, measure the similarity between different types of tree structure data, and verify the difference in the distribution of activated neurons in different types of input neural networks.

Table 1. Model samples the TSSID and TSSOD distance of 10 groups of ten kinds of tree structure data

Model	1	2	3	4	5	6	7	8	9	10
1	**1.58**	6.66	4.69	4.61	5.56	3.27	5.1	5.71	4.35	4.73
2	6.66	**1.97**	5.05	5.41	6.14	6.49	5.58	3.76	5.98	4.45
3	4.69	5.05	**2.39**	4.36	6.59	4.64	4.97	4.72	4.88	4.92
4	4.61	5.41	4.36	**2.3**	6.98	3.93	5.4	3.45	4.63	4.18
5	5.56	6.14	6.59	6.98	**2.53**	6.57	6.19	6.06	6.72	5.67
6	3.27	6.49	4.64	3.93	6.57	**2.4**	4.64	5.43	3.74	4.71
7	5.1	5.58	4.97	5.4	6.19	4.64	**2.39**	5.05	4.83	4.75
8	5.71	3.76	4.72	3.45	6.06	5.43	5.05	**2.2**	5.33	2.88
9	4.35	5.98	4.88	4.63	6.72	3.74	4.83	5.33	**2.3**	5.02
10	4.73	4.45	4.92	4.18	5.67	4.71	4.75	2.88	5.02	**2.26**

The data on the diagonal of the table is the internal distance (TSSID) of each category tree structure data, and the rest are the external distances (TSSOD) from different categories to other category tree structure data.

From the experimental data, as the complexity of the model increases, the difference between samples also increases. In addition, it can be clearly seen that the data on the diagonal line is significantly smaller than the other data in the column where it is located that is, we verify that

$$TSSOD(t_x, t_y) > TSSID(t_x, t_y), \forall x, y \in label \tag{10}$$

4.2 Multi-Layer Neuron Test Coverage Index Comparison Experiment

We compare the proposed Multi-Layer indicator with the existing test criteria, and verify the effectiveness of our indicator from the perspective of the overall structure coverage of the neural network.

In this paper, two sets of experimental designs are carried out, which are compared with the control indicators from the perspective of the full training set. Neuron Coverage (NC) in DeepXplore is used as the index of the comparison group, and all the

training set parts of the dataset MNIST are used, and a total of 60,000 sample pictures are all input into the model. The control group adopts the NC index as the coverage measurement index, and the experimental group adopts MLNC. Indicators, the test objects are set to three models of LeNet1, LeNet-4 and LeNet-5, take the average value of the final indicators of the three models as the result of the experiment and compare the changes of the indicators of the three models respectively, and observe the test criteria in the full amount. The variation on the training set samples, the sampling interval is numerical sampling every 1000 samples.

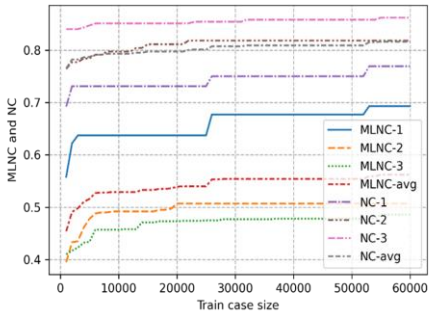

Fig. 3. Comparison of traditional white-box testing and white-box neural network-based testing

It can be seen from the figure that the NC index has reached a value of 0.765 in the initial stage, while the comparative MLNC-avg index only reached 0.454. After 60,000 experimental samples, the NC-avg index increased to 0.816, while the MLNC- avg index increased to 0.816. Raised to 0.562. From the experimental results, the MLNC index is initially lower than the NC index, indicating that the test space mapped by the MLNC index is larger than the NC index, and the same is true for the subsequent index increase. Therefore, the experiments in this section can verify that the MLNC indicator can measure the deep neural network in a more detailed manner. Compared with other test criteria, it requires more sufficient test case construction, and can conduct more detailed verification of the model decision logic.

Based on the three models of LeNet-1, LeNet-4 and LeNet-5, compare the changes of the experimental KMNC, NBC, SNAC and MLNC indicators after adding adversarial samples for many times. We select 10,000 random samples from the MNIST dataset as the base dataset for Test. On this basis, iteratively added a variety of adversarial samples for experiments, the first time adding 1000 FGSM adversarial samples, the second adding 1000 BIM samples, the third adding 1000 JSMA samples, and the fourth adding 1000 CW samples, and observe the changes of the test criteria on different models.

Table 2. Multi-layer neuron test coverage index different model experiments

DNN	Criteria	Test Org	Step 1 +FGSM	Step 2 +BIM	Step 3 +JSMA	Step 4 +CW
LeNet-1 (%)	KMNC	65.3	74.9	70.8	77.7	72.6
	NBC	43.5	47.3	49.2	45.4	44.3
	SNAC	35.3	42.5	47.3	44.2	42.1
	MLNC	**62.5**	**63.7**	**63.7**	**64.2**	**65.3**
LeNet-4 (%)	KMNC	71.4	74.2	74.5	77.6	76.3
	NBC	7.2	10.8	13.5	15.4	12.3
	SNAC	13.5	17.6	21.3	24.7	15.3
	MLNC	**48.2**	**51.1**	**51.1**	**51.3**	**52.3**
LeNet-5 (%)	KMNC	56.3	68.6	70.4	75.4	77.3
	NBC	5.6	13.3	16.5	35.0	45.5
	SNAC	13.9	18.9	19.4	22.4	14.5
	MLNC	**45.7**	**46.7**	**46.7**	**48.3**	**48.8**

In the three models of LeNet-1, LeNet-4 and LeNet5, as the model structure becomes more complex, the initial value of the MLNC index Test is lower, which reflects that MLNC is a reflection of the complexity of the model. The more complex the structure, the more logical structure needs to be tested. After the adversarial samples are added in stages, the MLNC indicator shows a relatively low growth rate compared with other test criteria, and the changes of other indicators exceed the MLNC indicator, which shows that in the case of the same amount of test data, the MLNC indicator is better than other indicators. The growth rate of the index is low, and the test and verification of the model is more stringent, which also proves the more detailed characteristics of the MLNC index.

4.3 Multi-Layer test case generation method comparison experiment

The basic process of the experiment is to first input the entire MNIST training set into the LeNet-1, LeNet-4 and LeNet-5 models of the three tested objects, record the final MLNC indicators and layer neuron activation states, and then use the best model in experiment three. The test case set generated by the experimental combination results is input to the MLNC index and layer neuron activation state recorded by the model, and the TTMLNC index and the TNMLNC index are calculated from this. The three indexes are used to measure whether the test case generation method is sufficient.

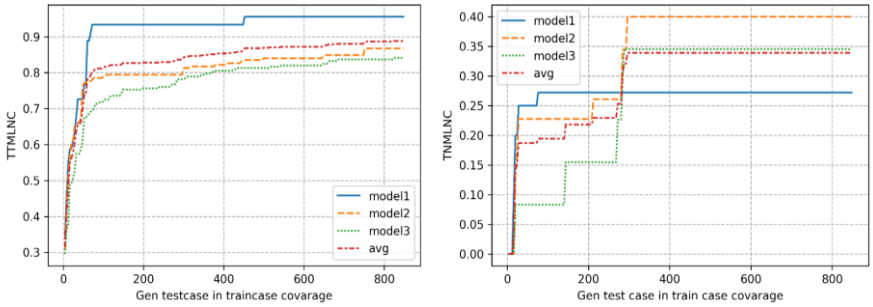

Fig. 4. TTMLNC and TNMLNC in 800 Generated test case

The adequacy experiment of the Multi-Layer test case generation method is constructed from the three aspects of MLNC index, TTMLNC index and TNMLNC index, and the quality of the test set is verified from the specific experimental results. The test case set has a comprehensive coverage rate of 64.4% in the three models, compared with the training set, the coverage is increased by 6.2%, covering 89% of the state space of the training set, and activating an additional 34% of the state space, and the verification test is sufficient.

5 Conclusion

We propose an algorithm to map a deep neural network into a tree structure, and use a related tree structure distance algorithm to measure the distance of the tree structure mapped by two test samples, and then quantify its change to the internal nodes of the deep neural network. The test method is improved from the white-box point of view, and relevant experimental indicators are constructed to verify the correctness of the algorithm. We experimentally verify that the deep neural network model has certain structural differences for different types of inputs. Therefore, a Multi-Layer test metric (MLNC) based on neural network structure is proposed, which can evaluate the state of the model from the perspective of model structure. TTMLC is proposed to evaluate whether the generated test cases can replace the existing test cases in terms of neuron coverage. TNMLNC is proposed on the neuron coverage to evaluate how many additional activated neurons the generated test cases are newer than the training data set.

Compared with the original training set of 10,000 samples, the generated 800 test cases can cover nearly 89% of the training set activation state on average on the three models, and an additional 34% of the training set activation state can be activated.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Zheng, W., et al.: An empirical study on correlations between deep neural network fairness and neuron coverage criteria. IEEE Transactions on Software Engineering **50**(3), 391–412 (2024)
2. Westhofen, L., Neurohr, C., Koopmann, T., et al.: Criticality metrics for automated driving: A review and suitability analysis of the state of the art. Archives of Computational Methods in Engineering **30**(1), 1–35 (2023)
3. Pei, K., Cao, Y., Yang, J.: DeepXplore: Automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 1–18 (2017)
4. Sun, Y., Huang, X., Kroening, D.: Testing deep neural networks. arXiv preprint arXiv:1803.04792 (2018)
5. Ma, L., Juefei-Xu, F., Zhang, F.Y., et al.: DeepGauge: Multi-granularity testing criteria for deep learning systems. In: Proceedings of the Automated Software Engineering, pp. 120–131 (2018)
6. Ma, L., Juefei-Xu, F., Xue, M., et al.: DeepCT: Tomographic combinatorial testing for deep learning systems. In: Proceedings of the 26th International Conference on Software Analysis, Evolution and Reengineering, pp. 614–618. IEEE (2019)
7. Szeliski, R.: Deep Learning. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-34372-9_5
8. An, S., Lu, R., Zhang, T.: Unsupervised feature learning for data classification. Journal of Physics: Conference Series **1994**, 012010 (2021). https://doi.org/10.1088/1742-6596/1994/1/012010
9. Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine learning testing: Survey, landscapes and horizons. arXiv preprint arXiv:1906.10742 (2019)
10. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3642–3649 (2012)
11. Hinton, G., Deng, L., Yu, D., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine **29**(6), 82–97 (2012)
12. Huval, B., Wang, T., Tandon, S., et al.: An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716 (2015)
13. Ciresan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in Neural Information Processing Systems (NIPS), pp. 2843–2851 (2012)
14. Chen, H., Engkvist, O., Wang, Y., et al.: The rise of deep learning in drug discovery. Drug Discovery Today **23**(6), 1241–1250 (2018)
15. Huang, X., Kroening, D., Kwiatkowska, M., et al.: Safety and trustworthiness of deep neural networks: A survey. arXiv preprint arXiv:1812.08342 (2018)
16. Xiang, W., Musau, P., Wild, A.A., et al.: Verification for machine learning, autonomy, and neural networks survey. arXiv preprint arXiv:1810.01989 (2018)
17. Offutt, A.J., Untch, R.H.: Mutation 2000: Uniting the orthogonal. In: Mutation Testing for the New Century, pp. 34–44 (2001)
18. Sun, Y., Huang, X., Kroening, D.: Testing deep neural networks. arXiv preprint arXiv:1803.04792 (2018)
19. Paaßen, B.: Revisiting the tree edit distance and its backtracing: A tutorial. arXiv preprint arXiv:1805.06869 (2018)

20. Tai, K.C.: The tree-to-tree correction problem. Journal of the ACM **26**(3), 422–433 (1979)
21. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing **18**(6), 1245–1262 (1989)
22. Pawlik, M., Augsten, N.: Tree edit distance: Robust and memory-efficient. Information Systems **56**, 157–173 (2016)
23. Pawlik, M., Augsten, N.: Efficient computation of the tree edit distance. ACM Transactions on Database Systems **40**(1), 1–40 (2015)
24. Khan, M.: Different approaches to black box testing technique for finding errors. International Journal of Software Engineering & Applications **2**(4), 1–12 (2011)