A Coverage-Guided Fuzzing Framework based on Genetic Algorithm for Neural Networks

Gaolei Yi
School of Reliability and Systems Engineering
Beihang University
Beijing, China
ygl666@ buaa.edu.cn

Xiaoyu Yang
China North Vehicle Research Institude
Beijing, China
15210280054@163.com

Pu Huang
School of Reliability and Systems Engineering
Beihang University
Beijing, China
p-huang@foxmail.com

Yichen Wang
School of Reliability and Systems Engineering
Beihang University
Beijing, China
wangyichen@buaa.edu.cn

***Abstract*—Due to the inherent difference between neural network and traditional software, it is very difficult to test it. At present, the use of fuzzing methods may be an effective exploration direction. We choose coverage-guided fuzzing as a method to test neural networks, and use neuron coverage as a coverage metric during execution. The effectiveness of neuron coverage will be demonstrated through experiments. On this basis, we designed a genetic algorithm-based fuzzing framework for neural networks, attempting to achieve greater coverage in a shorter time. And through the method of experimental comparison, the test efficiency of the framework is verified.**

Keywords: neural network; fuzzing; genetic algorithm

I. INTRODUCTION

With the development of related technologies in the field of neural network (NN), deep learning (DL) technology has been developed by leaps and bounds, and has been widely used in many fields such as computer vision and natural language processing.

Deep learning technology has gradually been applied to safety-critical areas, such as autonomous driving and assisted medical care. However, research in recent years has also shown that the training process of artificial intelligence technology is very susceptible to disturbances and may fail under relatively small adversarial interference. Although it is possible for learning algorithms to obtain generalization capabilities from data, their uncertainty has caused people to worry about their applications in safety-critical fields such as autonomous driving. Because of an input disturbance that humans cannot recognize, the learning algorithm may react in the opposite direction.

The whole life cycle of a machine learning system including the process of machine learning testing includes: obtaining a preliminary model from historical data, and conducting off-line testing (such as cross validation) on the model before the actual deployment of the system to ensure that the model meets the required requirements. After the system is formally deployed, the model starts to perform tasks such as prediction and decision-making. In this process, it generates new data that can be analyzed through online testing to evaluate how the current model affects user behavior [15]. Compared with online testing, offline testing has some limitations, but offline testing is essential for the initial evaluation of the system. In our current research, the neural network oriented testing we discussed mainly refers to offline testing, and our experimental research is also the category of offline testing.

The defect that causes the current behavior of the machine learning system to be inconsistent with the expected behavior is called a machine learning defect, and the activity of detecting machine learning defects is called a machine learning test. However, machine learning systems and traditional software systems are quite different in principle and structure, and machine learning testing faces many challenges. On the one hand, machine learning uses data-driven modeling to complete prediction or decision-making tasks. This data-driven modeling makes the system behavior change as the training data changes; on the other hand, the statistical nature of the model makes the output of the system uncertain. It is difficult to find test predictions. In addition, a machine learning system usually includes data (training set, verification set, test set), learning program (code written by developers to build and verify the model), implementation framework (code library and platform for building machine learning model). The effect is a composite effect, and it is unreasonable to split the system into multiple components to test separately.

The common practice of existing deep learning testing is to collect as much real data as possible and manually label it, or to generate a large amount of simulation data. But this method can only cover a small part of the real situation, and pay more attention to the coverage of the application situation, rather than paying attention to whether the test data, fed to the system, can cover most of the system logic; secondly, the method of manually labeling the data is time-consuming and laborious. Our experience with traditional software shows that it is difficult to build robust security critical systems only by using manual test cases. Considering the above reasons, we hope to use a method which is good at generating new test data

2767-6684/21/$31.00 ©2021 IEEE
DOI 10.1109/DSA52907.2021.00054

and does not need to label the data manually. After years of development, fuzzy testing has become one of the main methods to generate test data and test force. The strategy of generating test data is divided into generation based and mutation based. The fuzzy test based on generation needs to generate new input according to the rules and standards of system input. The fuzzy test based on variation generates new test data on the basis of existing data according to the variation rules. The difference between the new data and the original data is strictly limited in a certain range. In theory, the quality of test data generated by the former is higher than that of the latter, but its disadvantage is poor applicability. Before generating test data, we need to have a comprehensive and thorough understanding of the rules and standards of the system, so when facing different objects under test, we need to redesign a set of methods to generate test data. The latter only needs to adjust the mutation strategy when facing new test objects. Secondly, due to the similarity between the generated data and the original data, the label of the test data generated by the variation based fuzzy test will not be changed, so the manual annotation of the data is omitted. Based on the above analysis and the study of other related research, we finally choose the fuzzy test method based on variation.

In our research, we hope to learn from the structural coverage in traditional software testing to ensure the adequacy of the proposed testing method. Based on the above problems, in this article we use coverage-guided fuzzing (CGF) to generate new test data on the basis of existing data. On the one hand, we hope to cover extreme cases that rarely occur in real situations, and on the other hand newly generated test data is labeled, which eliminates the tedious work of manually labeling the data.

Using the CGF method, we need to select an effective coverage-metric. After investigation and combining our requirements for easy calculation of the selected coverage metrics, we finally selected neuron coverage as the coverage metric, and according to the principle of CGF, algorithm needs to be designed to achieve higher coverage. When analyzing the execution process of CGF, we believe that the entire process from the selection of the initial seed to the maintenance of the seed corpus to achieve maximum coverage is very similar to the implementation process of genetic algorithms. Genetic algorithm is a method of selecting the optimal solution through continuous iteration in the solution space, and it is often used in academia and industry to solve optimization problems.

The contributions of this article include:

- Discuss the effectiveness of neuron coverage as a measure of deep neural network(DNN) test adequacy.
- A CGF framework is proposed (shown in
- Figure 1), which uses CGF combined with genetic algorithm to maximize the coverage of neurons by input use cases.

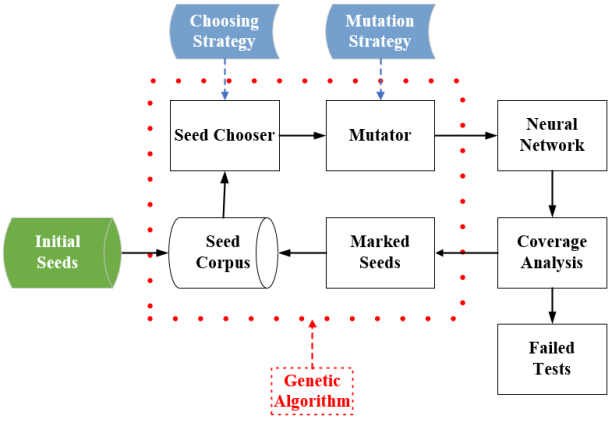

Figure 1. A CGF framework based on genetic algorithm

The structure of this paper is as follows: The second section introduces the possibility of CGF for DNN testing and the selection of DNN test coverage metrics. The third section introduces the CGF framework based on genetic algorithms for neural networks. The fourth section includes our experiments and comparisons. As a result, the final conclusions and prospects for future research directions are made.

II. BACKGROUND

This section gives the background basis for our research, including CGF for traditional software, and then discusses the feasibility of CGF applied to neural networks, and the selection of coverage metrics in the process of implementing CGF.

A. *Fuzzing-guided Fuzzing*

Fuzzing is a traditional automated testing technology that generates random data as program input to detect program crashes, memory leaks, etc. Due to its good scalability and effectiveness, it has been successfully applied to system security and vulnerability detection. [1]. As a method of fuzzing, CGF traditionally uses code lines and branches as coverage metrics, and uses code coverage metrics as feedback to maintain a balance between test effectiveness and test efficiency [2]. Many advanced CGF tools [4,5,6] have been widely used and proven effective.

At present, there have been many studies applying the CGF method to the test of deep learning [8, 9, 10]. They use different methods to achieve coverage improvement. On the basis of these studies, we believe that despite the huge differences between traditional programs and DNN, the success of CGF on the former still makes us think that using CGF to test the latter may have great potential. And some correspondences can be found in the category of CGF between traditional software and DNN. For example, the traditional tested program corresponds to DNN, the mutated seeds corresponds to the input of DNN, and the coverage feedback may also correspond to a kind of coverage of DNN. Of course, considering the uniqueness of DNN, its effective coverage

index is still worth discussing. This article also explores the effectiveness of neuron coverage.

B. Coverage metrics for DNN

A test adequacy metric, or a test coverage metric, is used to examine the adequacy of a set of test cases for testing the software under test under certain conditions [7]. Coverage in traditional software testing is related to software requirements and code coverage (such as statements, branches, conditions, etc.), but obviously these coverage criteria are not feasible in DNN testing, so we need to propose a new set of coverage criteria for DNN.

Neuron coverage is the first coverage index proposed for neural networks[11], which is defined as the proportion of activated neurons in the neural network. Neuron coverage has been applied to the study of DeepXplore. Attack resistant testing is a common deep learning testing method, but the change of the existing input is only limited to very small changes. If the change is too large, it needs to be manually annotated to determine the output of the system. Secondly, the test based on adversarial attack also does not consider whether the generated attack data can cover most of the system logic after entering the system. Therefore, the first contribution of DeepXplore is to propose a new test completeness measure neuron coverage for deep learning system testing. Deep Xplore is a deep learning system testing technology based on test case generation. The generated test input can not only expand the training set and retrain to improve the accuracy and robustness, but also detect the possible data pollution attacks in the training set. The specific method of detecting contaminated data is to train DNN for clean data set and contaminated data set respectively, and then generate new test input for these two DNNS with DeepXplore method on clean data set. The author thinks that the test input generated in this way is likely to be very close to the contaminated data. If we compare the similarity of some samples, we can find out the wrong labeled data in the contaminated data set.

On this basis, other coverage metrics including k-multisection neuron coverage[12] and ss-coverage [7] have been proposed. The author of deepGauge thinks that the decision-making behavior of DNN model is mainly determined by the connection weight between neurons and the activation function on neurons. The new test adequacy criterion can not only rely on the decision output of DL system, but should monitor and measure the neuron activity (nonlinear activation) and network connection (connection weight) from different granularity levels. Moreover, the neuron coverage proposed by deepxplore is more inclined to test the main functional areas, and rarely covers the extreme cases. How to cover more extreme cases is the key problem to be considered in the future DL test method design. Therefore, the author proposes a series of test criteria for multi-level and multi granularity coverage.

Through the study of multiple coverage metrics, we believe that other coverage metrics are improved on the basis of neuron coverage, and neuron coverage can be regarded as the basis. At the same time, there is a similar view in the standard document ISO / IEC TR 29119-11 [17], that is, if other coverage metrics are fully achieved, the neuron coverage will be automatically achieved.

In addition, because the machine learning system in the real use of time is relatively short, there is no complete test system for machine learning system, many problems are still worth discussing. For example, whether the white-box coverage metric of neural network is really effective or not is controversial in the academic circles. In addition to the above-mentioned studies, there are also related studies that question the coverage metrics such as neuron coverage [18] [19]. Therefore, we think that since we want to choose the neuron coverage as the coverage metric, we need to preliminarily verify the effectiveness of the neuron coverage on the tested neural network.

Here, we think that to discuss the effectiveness of the coverage metric, we should start from the basic coverage metric, just like the statement coverage in traditional software testing. At the same time, in order to realize the operation of the whole framework as soon as possible, we choose the neuron coverage as the DNN test coverage metric in this paper. We hope to discuss the relationship between the neuron coverage of test cases and the correct rate of prediction results. Because we traditionally believe that the higher the coverage achieved in the test process, the greater the possibility of finding problems. If this conclusion is valid for neural network, then we can achieve a better realization that when testing the neural network high neuron coverage is the goal of the test method, which also conforms to the principle of CGF. It only needs to achieve a higher neuron coverage to show that the test of the neural network is adequate.

III. METHODOLOGY

In this section, we will introduce the proposed CGF framework. Firstly, we will give an overview of the entire framework, and then describe key components in detail.

A. CGF Framwork

In this paper, based on the research of coverage guided fuzzing of traditional computer programs, we hope to propose a neural network oriented fuzzy testing framework based on genetic algorithm, which is universal and extensible when facing different systems under test. We introduce genetic algorithm to optimize the whole CGF process, and improve the efficiency of this method based on neuron coverage and seed selection strategy.

The CGF framework, as shown in Figure 1, starts with selecting seeds from initial seeds to generate a seed corpus according to the selection strategy, which corresponds to the initial population selection in the genetic algorithm. This strategy assigns a probability to each seed based on the number of times each seed has been selected and whether a new coverage was exercised in the previous iteration. When selecting the seed, the seed with the higher probability is preferentially selected according to the probability. The detailed probability calculation method in this process will be described in III.B.

After obtaining the seed corpus, that is, the population in the genetic algorithm, the seed is not the real input. The mutator needs to mutate the seeds according to the mutation

strategy, and obtain the new input as the input of the target program to make the program run. The mutation strategy can be flipping a pixel of the input image, or it can be a limited modification of the seeds according to a certain constraint. The mutation strategy will be described in detail in III.C.

The mutated seed will eventually be fed to the neural network. During the execution of the neural network, we will monitor the coverage of neurons and the output results. In this framework, two parts will be obtained from the neural network, including a neuron coverage sequence, from which the neuron coverage and the output of the neural network can be calculated.

After calculating the coverage, which is corresponded to the fitness function in genetic algorithm, we can mark the interesting seeds which exercises new coverage and add them to the seed corpus. The above process will continue to iterate until the stop condition is met. According to the above description, it is not difficult to find that the whole process of the framework is the same as that of CGF. The difference lies in the formulation of seed selection strategy and the overall planning of the testing process by using genetic algorithm, selecting the appropriate seed mutation strategy to mutate the seeds, and marking the seeds of interest.

The key component in this iterative process is seed selection strategy and mutation strategy. We regard the entire iterative process as an iterative process of a genetic algorithm, and the ultimate goal is to find the optimal solution to maximize coverage.

B. Seed Selecting Strategy

In each iteration of CGF, a fixed number of seeds must be selected from the seed corpus (translation note) as the original input before mutation. In order to achieve high coverage faster, we need to sort the seeds in the queue. The seeds that are more likely to exercise a new coverage rate (translation attention) have a higher probability of being selected. Therefore, the probability of our seed being selected meets the following characteristics:

1. The probability value can be guaranteed to be between (0,1);
2. The more times a seed is selected, the lower the probability of being selected again.

Therefore, we believe that the following function can meet our requirements:

$$P_1(\mathrm{x}) = \frac{1}{1+e^{ax+b}}\ , a > 0, x \in [0, +\infty) \qquad (1)$$

where x is the number of times the seed has been selected.

On the other hand, in the middle and late stages of the test, the new neuron coverage generated by the test input will become very small, and it is these very few new neuron coverages that are extremely precious to our test, because these new coverages increase in the rate means that the possibility of finding erroneous behaviors is greater. So we designed another part of the probability function for each seed:

$$P_2(NCov_{new}, Circle, Total) = \sqrt[1+\frac{Circle^4}{Total^3}]{NCov_{new}} \qquad (2)$$

where $NCov_{new}$ is the new coverage that appeared in the last iteration test, Circle is the current iteration number, and Total is the target iteration number. In this function, when the Circle is small, it is similar to a linear function, which can well reflect the discrimination of the new coverage index; when the Circle is large, this function can increase the impact of the small new coverage on the seed probability.

In the end, the probability of each seed being selected is:

$$\mathrm{P} = \left(1 - \frac{Circle}{Total}\right) \times P_1 + \frac{Circle}{Total} \times P_2 \qquad (3)$$

As the test progresses, the proportion of P_2 will be higher, that is, the seeds that exercise new coverage are more likely to be selected. In addition, in the process of maintaining the seed corpus, there is a certain probability that seeds that have never been selected from the initial seeds are selected to improve the mutation efficiency in the genetic algorithm.

In the process of mutating the seed image, we hope that the mutated image has no obvious difference in the eyes of humans. This is also the real test case generated by the fuzzer in the fuzzing test. Traditional image blur operations include random changes in pixel values, image rotation, and changes in image brightness. However, some of these methods, such as image rotation, cannot meet our requirement that the mutated image is not significantly different from the human eye. So we finally choose two methods as image mutation methods: Gaussian noise with configurable mean and standard deviation and inversion of random pixel value.

C. Mutation Strategy

In the process of mutating the seed image, we hope that the mutated image has no obvious difference in the eyes of humans. This is also the real test case generated by the fuzzer in the fuzzing test. Traditional image blur operations include random changes in pixel values, image rotation, and changes in image brightness. However, some of these methods, such as image rotation, cannot meet our requirement that the mutated image is not significantly different from the human eye, so we finally chose to add Gaussian noise with configurable mean and standard deviation and random pixel values. The inversion of the two methods is used as the means of image mutation. Take an image in the MNIST dataset as an example

Gaussian noise, as the name suggests, is a kind of noise that obeys Gaussian distribution. Adding Gaussian noise to the image will make the probability density function of the new image noise obey Gaussian distribution. In the field of image processing, Gaussian noise belongs to additive noise, and new image is the superposition of noise and original image.

Pixel inversion is generally applied to gray image. Each pixel of gray image is represented by a value between (0, 255). 255 represents white and 0 represents black. Pixel flipping is to transform the selected pixels so that the values before and after the transformation are 127.5 symmetrical with respect to

the points on the number axis, which is called "black and white inversion".

The result after being mutated by the mutator according to the mutation strategy is shown in Figure 2. In the process of research, we find that the same image, which is obtained by adding Gaussian noise and pixel inversion, is more likely to lead to the error behavior of the tested neural network. So we set the probability of pixel inversion higher: for each mutation, there are 30% and 50% probability of Gaussian blur and pixel inversion separately, and another 20% probability of both Gaussian blur and pixel inversion.

Figure 2. The original picture and the picture after adding Gaussian noise and pixel inversion respectively

IV. EXPERIMENTS

In response to the above content, we have raised two corresponding questions, which need to be demonstrated experimentally.

RQ1: Is neuron coverage effective as a coverage metric of neural network test adequacy?

RQ2: Compared with the strategies in other papers, does the seed selection strategy we designed have an advantage in improving coverage?

A. Dataset and CNN Models

We chose the commonly used public data set MNIST as the experimental data set. MNIST is a data set for handwritten digital image recognition, including 60,000 training data and 10,000 test data, a total of 70,000 images, divided into 10 categories from 0 to 9. Each MNIST image is single-channel and the size is 28×28×1. We trained different CNN models as subject models on the MNIST data set. The training process of each model is carried out under the same controllable conditions, such as training data, number of training iterations, etc.

B. Effectiveness of Neuron Coverage

We have chosen neuron coverage as the coverage metric, which means that the hypothesis is that the higher the neuron coverage reached, the more incorrect prediction results of the CNN model will be. When the coverage is high enough, the test will be adequate. In order to verify this hypothesis, a response experiment needs to be designed to prove it.

1) Experiment Setup

In order to answer RQ1, we feed test cases to different CNNs, record the neuron coverage and the number of false predictions during the test, and then analyze the correlation between the neuron coverage and the number of false predictions based on statistical data.

2) Objective of the Experiment

Experiments are designed to prove the effectiveness of selecting neuron coverage as the coverage index, that is, to verify the correlation between the coverage and the number of erroneous behaviors.

3) Experimental Hypothesis

The purpose of this experiment is to verify the effectiveness of using neuron coverage as an evaluation index of fuzzy test in subsequent experiments. Therefore, it is necessary to construct and verify the relationship between neuron coverage and an index that directly reflects the effectiveness of fuzzy test. The most direct indicator to reflect the effect of fuzzy test is the number of erroneous behaviors detected. Therefore, the hypothesis of this experiment is: in the same neural network, the neuron coverage index after the test run is positively correlated with the number of erroneous behaviors detected.

4) Experimental Model and Experimental Data

We choose MNIST as the experimental data set. MNIST is a data set for handwritten numeral image recognition, including 60000 training data and 10000 test data, a total of 70000 images, divided into 10 categories from 0 to 9. Each MNIST image is a single channel with a size of 28×28×1.

We train different CNN models as subject models on MNIST dataset. The training process of each model is carried out under the same controllable conditions, such as training data, training iterations and so on.

5) Experimental Variables

The independent variable of this experiment is the coverage rate of neurons, and the dependent variable is the number of error behaviors detected.

6) Experiment Results

In the test process of the first CNN, we set the number of each test set to be the same. The statistical results are shown in Figure 3. It can be seen that when the coverage of the test set increases, the number of erroneous results predicted by CNN increases. At this time, the correlation coefficient of the two variables is 0.656, and the significance p-value is 0.0392, indicating that the two have a certain positive correlation.

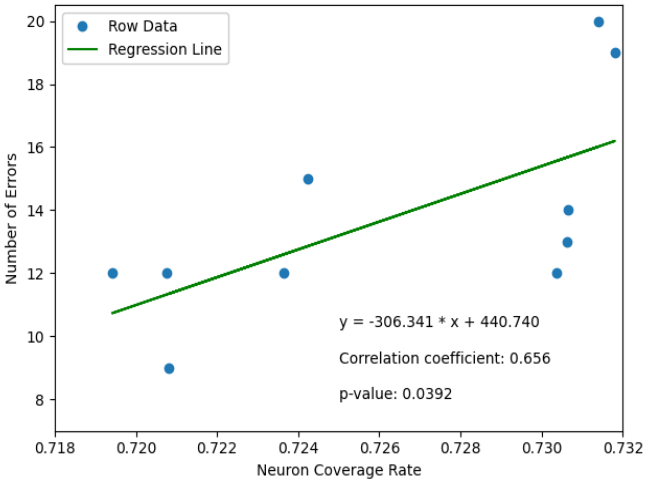

Figure 3. The relationship between neuron coverage and the number of erroneous behaviors on CNN-1

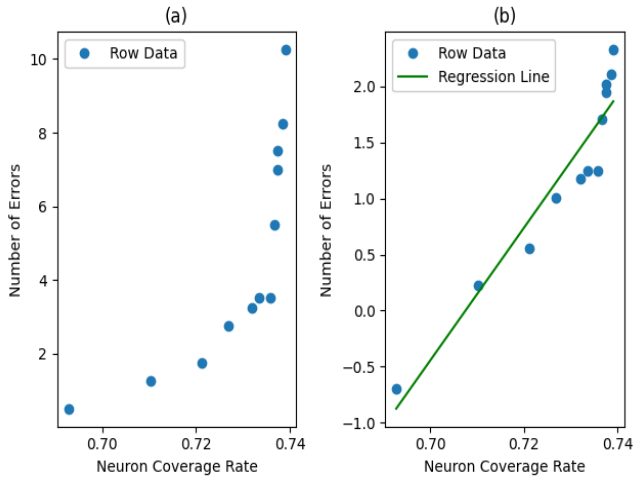

Figure 4. The relationship between neuron coverage and the number of erroneous behaviors on CNN-2

In the second CNN test process, we chose different size test sets. The statistical results are shown in Figure 4. It can be seen that when the coverage rate increases, the number of prediction erroneous behaviors increases in a manner that approximates exponential growth. After taking the logarithm to do a linear fitting, the correlation coefficient at this time is 0.950, and the significance p-value is $2.18 \times 10^{-6} \ll 0.01$.

Based on the above analysis of the statistical results, it can be concluded that increasing the coverage of neurons to a certain extent can help us find more potential erroneous behaviors in the deep learning model.

C. Comparison of test efficiency of CGF framework

1) Objective of the Experiment

Through comparative experiments, the efficiency of the proposed seed selection strategy based on Genetic Algorithm in coverage will be analyzed.

2) Experimental Hypothesis

In the same test environment, the performance of our optimization strategy is better. That is, in the same time, the coverage can be higher.

3) Experimental Environment

The experimental environment is the neural network construction, coverage calculation principle and image mutation mode described in the previous sections. It is worth mentioning the construction of the new coverage index calculation module.

The function of the new coverage calculation module, as the name suggests, is to calculate the new coverage generated after the image is input into the neural network. Here we want to define the calculation method of the new coverage rate: the new coverage rate in this paper refers to the proportion of the number of neurons not covered in the current round to the number of neurons not covered in the previous round, as shown in the following formula:

$$\mathrm{NewNCov}(\mathrm{net}, \mathrm{n}) = \frac{net.NewActivated(n)}{net.NotActivated(n-1)} \qquad (4)$$

where n is the number of iterations.

4) Experiment Setup

In order to answer RQ2, we designed an experiment to compare the proposed seed selection strategy with random selection and the seed selection strategy proposed in other papers [8, 9], and compare the neuron coverage achieved by the four after the same number of test iterations.

5) Experiment Results

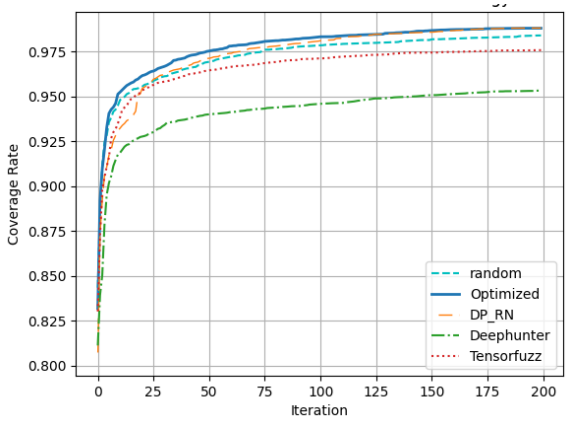

Figure 5. Comparison of the efficiency of seed selection strategies

As shown in Figure 5, DP_RN is an improved method based on the seed selection strategy proposed by DeepHunter. This method uses random selection in the early stage of the test, because we find that random selection has great advantages in the early stage of the test. Based on this finding, in our proposed seed selection strategy, a random selection method was also used in the early stage to improve efficiency. It can be seen from the figure that after the first few iterations, the neuron coverage achieved by our proposed method is relatively higher.

V. Conclusion and Future Works

Deep learning has achieved great success in the past decade and has become the main driving force driving the development of many new intelligent applications. However, the quality assurance technology of deep learning-based systems is still in its early stages and requires a very scalable test framework. In this article, we propose a genetic algorithm-based fuzz testing framework for neural networks, select neuron coverage as a measure of test adequacy, and combine the advantages of genetic algorithm in optimization calculations, hoping to achieve it in the shortest time Higher neuron coverage. By designing experimental verification methods, we explained the effectiveness of neuron coverage as a measure of test adequacy. On this basis, we have improved our method based on the experimental results and further improved its efficiency by comparing experiments with other strategies.

It is worth noting that although we use the experimental method to verify the effectiveness of neuron coverage, we think it can only show that there is a correlation between neuron coverage and neural network error behavior, rather

than causality. In view of the fact that there is still no unified understanding of the white box coverage index of neural network, and the neural network technology will encounter more challenges in the development process, we will also expand our proposed framework in the future work combined with the existing coverage index.

In future work, we will make further improvements to the proposed fuzzing framework, try to use other improved coverage metrics coverage criteria, try other algorithms other than genetic algorithms to further improve the test efficiency, and will use more complex data sets such as ImageNet in Experiments. At the same time, we hope to further explore the specific representation of the input that causes the wrong prediction result in the neural network, because we think this will help us to further improve the test method.

REFERENCES

[1] Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz,and Shin Yoo. The oracle problem in software testing: A survey.IEEE transactions on software engineering, 41(5):507–525, 2015

[2] Mike Aizatsky, Kostya Serebryany, Oliver Chang, Abhishek Arya, and Meredith Whittaker. Announcing oss-fuzz: Continuous fuzzing for open source software. Google Testing Blog, 2016.

[3] Augustus Odena and Ian Goodfellow. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. *arXiv preprint arXiv:1807.10875,* 2018.

[4] "American Fuzzy Lop," 2018. [Online]. Available: http://lcamtuf.coredump.cx/afl/

[5] "libFuzzer," 2018. [Online]. Available: https://llvm.org/docs/LibFuzzer.html

[6] S. Rawat, V . Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos, "Vuzzer: Applicationaware evolutionary fuzzing," in Proceedings of the Network and Distributed System Security Symposium (NDSS), 2017.

[7] Y. Sun, X. Huang, and D. Kroening. Testing Deep Neural Networks. ArXiv e-prints, March 2018

[8] Odena, A. and Goodfellow, I., "TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing", arXiv e-prints, 2018.

[9] Xie, X., "DeepHunter: Hunting Deep Neural Network Defects via Coverage-Guided Fuzzing", arXiv e-prints, 2018.

[10] Guo, J., Jiang, Y., Zhao, Y., Chen, Q., and Sun, J., "DLFuzz: Differential Fuzzing Testing of Deep Learning Systems", arXiv e-prints, 2018.

[11] Pei K, Cao Y, Yang J, et al. Deepxplore: Automated whitebox testing of deep learning systems. In: Proc. of the 26th Symp. on Operating Systems Principles. ACM, 2017. 1−18.

[12] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. DeepGauge: Multi-granularity Testing Criteria for Deep Learning Systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, pages 120–131, 2018.

[13] Xiang Gao, Ripon K. Saha, Mukul R. Prasad, and Abhik Roychoudhury. 2020.Fuzz Testing based Data Augmentation to Improve Robustness of Deep Neural Networks. In 42nd International Conference on Software Engineering(ICSE '20).

[14] J. Guo, Y. Zhao, Y. Jiang and H. Song, "Coverage Guided Differential Adversarial Testing of Deep Learning Systems," in IEEE Transactions on Network Science and Engineering. doi: 10.1109/TNSE.2020.2997359

[15] J.M.Zhang, M.Harman, L.Ma and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2019.2962027.

[16] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing advanced driver assistance systems using multiobjective search and neural networks. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pages 63–74. ACM, 2016.Zalewski, M (2017) American fuzzy lop. http://lcamtuf.coredump.cx/afl/.Accessed 25 Dec 2017.

[17] ISO/IEC TR 29119-11:2020 Software and systems engineering — Software testing — Part 11: Guidelines on the testing of AI-based systems.

[18] Harel F Y. Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks?[J]. Ann Arbor, 1001: 48106-1346.

[19] Dong Y, Zhang P, Wang J, et al. There is Limited Correlation between Coverage and Robustness for Deep Neural Networks[J]. arXiv preprint arXiv:1911.05904, 2019.