Test Input Selection for Deep Neural Networks

Zhiyu Wang¹, Sihan Xu*², Xiangrui Cai¹, Hua Ji¹

¹College of Cyber Science, Nankai University, Tianjin 300071, China
²College of Computer Science, Nankai University, Tianjin 300071, China

wangzhiyu@mail.nankai.edu.cn, {xusihan, caixr, hua.ji}@nankai.edu.cn

Abstract. With the rapid development of deep learning technologies, the quality and security of deep learning systems have aroused great concern recently. Much research has been done in testing deep learning systems. Nevertheless, the oracle problem still remains since the input space of DNN-based software is usually very large and manually labeling is boring and cost-effective. Inspired by structural testing for traditional software, in this paper, we propose an algorithm to mitigate the oracle problem by selecting test inputs worth labeling. Specifically, we propose a test subset selection algorithm that can automatically select a test suite with high coverage but a small size when the labeling budget is limited. Compared with DeepXplore, the proposed algorithm can generate smaller test suites with higher coverage.

1. Introduction

Deep Neural Networks (DNN) have gained great popularity in various intelligent applications over the last decade, such as image processing, speech recognition, and autonomous driving. However, despite the tremendous progress, like traditional software, DNN-based software often exhibits unexpected behaviors when deployment in real-world applications, such as self-driving car accidents [4] and interference with Google’s face recognition system by blocking and changing the brightness [3]. These erroneous behaviors have caused concerns about the safety and quality of deep learning systems.

Although deep learning testing has attracted much attention recently, to test all possible inputs is impractical. On the one hand, the input data of deep learning systems is often continuous and high dimensional. On the other hand, testing deep neural networks requires the ground truth of testing data, which is mainly based on manually labeling. Nevertheless, manually labeling the testing suite takes a lot of time and often requires professional knowledge. Therefore, with a limited labeling budget, selecting labeling-worthy test inputs from a large-scale pool of unlabeled inputs is of great significance.

In this paper, we propose a test selection algorithm based on structural coverage criteria to mitigate the oracle problem of deep learning testing. The basic intuition is that test sets with higher structural coverage criteria are ought to have higher diversities. From a large-scale pool of unlabeled test inputs, the goal of this paper is to select a representative test suite that is worth the labeling effort. Our main contributions are as follows:

1) We propose a test subset selection algorithm based on dynamic programming and structural coverage criteria specified for DNNs. In each iteration, the smallest subset that can achieve the highest coverage is selected from the unlabeled test set. Experiments on the two public datasets show that the proposed method achieved higher coverage with smaller test suite than DeepXplore.

2) Based on the proposed algorithm, we also designed a sorting algorithm to prioritize test inputs. By sorting the unlabeled test set according to the error detection ability, we have achieved a 94% error sample recall rate under the 10% scale of the original test set.
2. Related Work

Inspired by the success of the coverage standard in testing traditional software systems, Pei et al. [5] first proposed Neuron Coverage, a DNN coverage criterion. However, Neuron Coverage is too simple to be satisfied, and it is difficult to effectively distinguish the adequacy of testing in complex models. Therefore, Ma et al. [1] put forward a variety of more complex structural coverage criteria. On this basis, Xie et al. [6] and Tian et al. [2] applied the current structural coverage criteria on image recognition and automatic driving and other fields and achieved good results.

Research efforts on testing for deep neural networks mainly focus on generating test case sets through mutation methods. According to the mutation method, it is divided into two kinds: 1) method based on fuzzing test (DeepHunter [6], TensorFuzz [7], etc.); 2) method based on neuron gradient (DeepXplore [5]). The fuzzing method uses a large number of random mutations and screening mechanisms to finally retain samples that can meet the conditions (such as increasing current coverage or causing the DNN to produce wrong behavior). The neuron gradient-based method directly uses the structural coverage as the objective function, and iteratively changes the original input data through backpropagation to generate a confrontation example that can generate new structural coverage or make the neural network show wrong behavior. However, these mutation methods are difficult to logically guarantee that the generated samples can maintain the assumption that the semantics are unchanged.

3. Methodology

In this section, we provide a detailed technical description of our algorithm. First, we introduce popular DNN software structure coverage criteria which we applied in our article. Second, we propose an optimal test subset selection algorithm based on the guidance of structural coverage. Finally, based on the optimal test subset selection algorithm, we design a sorting algorithm to prioritize test inputs.

3.1. Coverage criteria for Deep Neural Network

The idea of DNN structural coverage criteria is inspired by white box testing in traditional software testing. Although the program structure of deep neural networks is different from the logical structure of traditional software, same as logical unit in traditional software, every neuron in the deep neural network structure often contains certain semantic information. If an input set can touch more neuron structures (active state and inactive state), we can intuitively think that it traverses more logical information contained in the current neural network.

Existing studies have proposed several neuron structure coverage fitness functions. Referring to the conclusions of DeepHunter and DeepGauge, we use the following 5 coverage criteria which are more commonly used and have better performance:

- **Neuron Coverage (NC).** NC bisects a neuron’s state into activated and non-activated. Given an input, a neuron is activated if its output value is above a predefined threshold. NC measures the ratio of activated neurons of a DNN.
- **$K$-Multisection Neuron Coverage (KMNC).** For each neuron, the range of its values (obtained from training data) is partitioned into $k$ sections. An input covers a section of a neuron if the output value falls into the corresponding value section range. KMNC measures the ratio of all covered sections of all neurons of a DNN.
- **Neuron Boundary Coverage (NBC).** Similar to KMNC, NBC analyzes the value range of a neuron covered by training data, and measures to what extent the corner-case regions outside the major functional range of a neuron are covered.
- **Strong Neuron Activation Coverage (SNAC).** Similar to NBC, for each neuron, SNAC considers the value range that is above the maximum value seen during training. SNAC measures how the upper corner-case regions of neurons are covered.
- **Top-$k$ Neuron Coverage (TKNC).** TKNC is a layer level testing criterion, which measures the ratio of neurons that have once been the most active $k$ neurons of each layer on a given test set.
3.2. Test selection algorithm

Given a deep neural network model $M$ and a structure coverage criteria $C$, for each sample $O_i$ in test set $O$, its structural coverage can be obtained according to the definition of structure coverage criteria. As a result, the minimum subset of the current test data set with the maximum coverage can be obtained through Algorithm 1.

**Algorithm 1** Optimal subset selection algorithm

**Input:** $M$:model, $N$:model's neuron set, $O$:test set, $C$:structure coverage criteria,

**Initialize:** $n$: Number of neurons under structure coverage criteria $C$  
$\Omega_i$: Neurons activated by test sample $O_i$  
$T$: The total number of neurons activated by test set $O$  
$N_{i,j}$: The most neurons that can be activated,  
Use $i$ of the first $j$ examples in the test set

**Output:** The smallest subset of $O$ that achieves the maximum coverage

1: for $i \leftarrow 0$ to $T$ do
2:    for $j \leftarrow 0$ to $n$ do
3:    if $N_{i,j-1} \geq (N_{i-1,j-1} \cup \Omega_j)$ then
4:        $N_{i,j} \leftarrow N_{i,j-1}$
5:    else
6:        $N_{i,j} \leftarrow N_{i-1,j-1} \cup \Omega_j$
7:    end if
8:    if $\text{sum}(N_{i,j}) = T$ then
9:        return $N_{i,j}$
10:   end if
11:  end for
12: end for

3.3. Test priority algorithm

Based on the proposed test selection algorithm, a representative test suite with high coverage but a small size can be generated from a large pool of unlabeled tests. In this section, we also propose an algorithm to further sort tests according to their importance. Specifically, given a large-scale unlabeled test set, we iteratively execute Algorithm 1, and then remove selected tests from the unlabeled test pool. In this way, a sequence of test inputs can be obtained according to the importance of tests. When the correlation between coverage criteria and error detection is considered, this sequence can be regarded as a test set ranking sequence based on error detection capability. The algorithm is described in Algorithm 2.

**Algorithm 2** Test set sample sorting algorithm

**Input:** $M$:model, $N$:model's neuron set, $O$:test set, $C$:structure coverage criteria,

**Initialize:** $D(P(O,M,N,F))$: Optimal subset selection algorithm  
$\Upsilon$: Sequence of the sorted test set

**Output:** Ordered test set

1: while $O \neq \emptyset$ do
2:    $\nu \leftarrow D(P(O,M,N,F))$
3:    $O \leftarrow O - \nu$
4:    $\Upsilon \leftarrow \Upsilon + \nu$
5: end while
6: return $\Upsilon$
4. Experimental Results

4.1. Test datasets and DNNs

To evaluate the effectiveness of the proposed algorithms, we conduct experiments on two popular public datasets, i.e., the MNIST and CIFAR-10 datasets. For each dataset, we exploit three well-known DNN models in the field of image classification. All DNN models are pre-trained to achieve comparable performance to conduct a fair evaluation. The details of datasets and models are as follows:

- **MNIST** is a handwritten digit dataset containing 28x28 pixel images with class labels from 0 to 9. The dataset includes 60,000 training samples and 10,000 testing samples. This paper uses two pre-trained LeNet family models (LeNet-1 and LeNet-5) to analyze the data set. The two models have 52 and 268 neurons respectively, which are classic small neural networks.

- **CIFAR-10** for large-scale DL system, CIFAR-10, is also selected in the experiment, which is a large set of conventional image classification dataset. CIFAR-10 contains 10 categories of 60000 images. The dataset is divided into five training batches and one test batch, each with 10000 images. Each picture has three channels with a size of 32 × 32 × 3. The DNN for CIFAR-10 is the pre-trained ResNet-20 and VGG-16 models, which are relatively large, and have achieved good results in common competitions.

4.2. Coverage comparison experiment

In order to test the performance of the algorithm in this paper, we first show the structure coverage criteria of the generated test set. DeepXplore is used as a baseline, which is a classic mutation test set generation method. We compared the average fitness of two methods under five different structure coverage criteria mentioned above. The average fitness is calculated by dividing the structure coverage of the generated test set by the number of samples it contains.

| Model    | Method     | NC   | KMNC | NBC  | SNAC | TKNC |
|----------|------------|------|------|------|------|------|
| LeNet-1  | DP         | 9.88 | 1.22 | 5.07 | 0.62 | 17.55|
|          | DeepXplore | 5.39 | 1.28 | 1.97 | 1.03 | 15.60|
| LeNet-5  | DP         | 12.23| 1.80 | 3.02 | 1.20 | 7.45 |
|          | DeepXplore | 8.15 | 2.22 | 1.98 | 2.82 | 7.45 |
| Resnet-20| DP         | 6.43 | 4.98 | 4.14 | 11.43| 9.51 |
|          | DeepXplore | 5.72 | 3.56 | 2.49 | 2.40 | 11.96|
| Vgg-16   | DP         | 25.10| 3.55 | 38.72| 14.15| 10.26|
|          | DeepXplore | 15.21| 3.95 | 2.82 | 2.02 | 11.86|

As shown in Table 1, the proposed method is superior to DeepXplore in most cases, although the maximum structure coverage achieved by the mutation-based DeepXplore may be higher than the DP algorithm, the number of samples in the generated test set for the DP method much less than DeepXplore.

4.3. Comparative experiment on error detection ability

In this section, we demonstrate the performance of the test set sorting algorithm with the error detection ability of its generated subset. For MNIST and CIFAR-10 dataset, we select top 10% samples from the test set to calculate the recall rate of wrong samples. The recall rate of wrong samples is defined as the number of error samples contained in the selected sample divided by the total number of error samples.

We conduct the experiments with a set of hyperparameters for different structure coverage criteria. We set $k = [0.5, 0.7, 0.9]$ in NC where $k$ is the threshold for judging whether the neuron is activated in NC cases. For TKNC, we set $k = [2, 3, 5]$, where $k$ indicates the number of the most active neurons extracted. The results of NC and TKNC are shown in Figure 1 and Figure 2:
It can be seen that in most cases NC and TKNC can reach the maximum coverage with only a small test suite, so in some cases, the performance of DP method is almost the same as the random method. For TKNC, the proposed algorithm achieved better performance than the random selection algorithm, while the recall rate of wrong samples for generating test set is still less than 50%.

We also found that our algorithm cannot adapt to the coverage criteria of neuron boundary. As shown in Figure 3, under the Neuron Boundary Coverage and Strong Neuron Activation Coverage, DP method is equivalent to the random selection method. Their recall rate of faults is close to 10%.

K-Multisection Neuron Coverage is the most stable structure coverage criteria in current research. We found that DP method can achieve an extraordinary error sample recall rate for different models when k is set to an appropriate value. We tried different values of k for LeNet-1 and LeNet-5 models. And in Figure 4(c), in the case of Resnet-20 and Vgg-16 models, k is set to 10.
It can be observed that the recall rates of error samples in DP method are higher than 94.1% for MNIST, and higher than 50% for CIFAR-10, which is much higher than random selection.

5. Conclusion
In this paper, we propose the test selection algorithm and the test priority algorithm based on structural coverage criteria. The goal of this paper is to select and prior important tests that are worth the labeling effort. Extensive experiments on two public datasets and four popular DNN models demonstrated that the proposed algorithm can achieve higher coverage with smaller test suits than DeepXplore. Moreover, experiments show that the proposed test priority algorithm is able to find more natural faults compared with random selection, achieving the recall rate of 94.1%.

Acknowledgments
This work was financially supported by “the Fundamental Research Funds for the Central Universities”, Nankai University (No. 63201166, 63201210 and No. 63201209)

References
[1] Ma, Lei, et al. "Deepgauge: Multi-granularity testing criteria for deep learning systems." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018.
[2] Tian, Yuchi, et al. "Deeptest: Automated testing of deep-neural-network-driven autonomous cars." Proceedings of the 40th international conference on software engineering. 2018.
[3] Google accident 2016: A google self-driving car caused a crash for the first time. http://www.theverge.com/2016/2/29/11134344/.
[4] Tesla accident 2016. Understanding the fatal Tesla accident on Autopilot and the NHTSA probe. https://electrek.co/2016/07/01/understanding-fatal-tesla-accident-autopilot-nhtsa-probe/. (2016).
[5] Pei, Kexin, et al. "DeepXplore: Automated whitebox testing of deep learning systems." proceedings of the 26th Symposium on Operating Systems Principles. 2017.
[6] Xie, Xiaofei, et al. "Deephunter: Hunting deep neural network defects via coverage-guided fuzzing." arXiv preprint arXiv:1809.01266 (2018).
[7] Odena, Augustus, and Ian Goodfellow. "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing." arXiv preprint arXiv:1807.10875 (2018).
[8] The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/
[9] The CIFAR-10 dataset. http://www.cs.toronto.edu/kriz/cifar.html