Summary on *DeepXplore: Automated White-box Testing of Deep Learning Systems*

Presented by : Ji Gao

[1]Department of Computer Science, University of Virginia
https://qdata.github.io/deep2Read/

August 26, 2018

DeepXplore: Automated White-box Testing of Deep Learning Systems

Abstract: . . . We design, implement, and evaluate DeepXplore, the first white-box framework for systematically testing real-world DL systems. We address two main problems: (1) generating inputs that trigger different parts of a DL systems logic and (2) identifying incorrect behaviors of DL systems without manual effort. First, we introduce neuron coverage for systematically estimating the parts of DL system exercised by a set of test inputs. Next, we leverage multiple DL systems with similar functionality as cross-referencing oracles and thus avoid manual checking for erroneous behaviors. We demonstrate how finding inputs triggering differential behaviors while achieving high neuron coverage for DL algorithms can be represented as a joint optimization problem and solved efficiently using gradient-based optimization techniques **. . .**

- It is important to ensure Deep Learning system works well. Need proper testing methods
- Problem on testing machine learning models:
  1. Input space is huge: clearly we can't test everything
  2. Lack of the oracle.
  3. The code is generated by machine
  4. It's a data-driven approach

Goal of this paper

This paper: use software testing skills to help.

- Input space is huge $\rightarrow$ define a new coverage metric
- Oracle problem $\rightarrow$ Pseudo oracle - N-version programming: Use multiple version of machine learning models

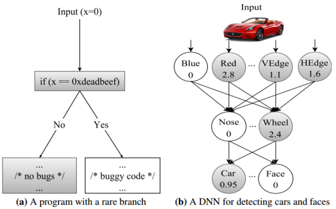


Figure 3: Comparison of program flows of a traditional program and a neural network. The nodes in gray denote the corresponding basic blocks or neurons that participated while processing an input.

Code coverage: a popular measure in software testing

However can't be directly used in Machine learning models. Every line of the code is always executed for any inputs.

Neuron coverage: how many neurons are activated (i.e., output value goes over a threshold) by the test inputs.

$$NCov(T, X) = \frac{|\{n | \forall x \in T, out(n, x) > t\}|}{|N|}$$

According to their pseudo-code, should be

$$NCov(T) = \frac{|\{n | \exists x \in T, out(n, x) > t\}|}{|N|}$$

Theory: each neuron in a DNN tends to be responsible for extracting a specific feature of the input instead of multiple neurons collaborating to extract a feature. [*Understanding Neural Networks Through Deep Visualization*] (Questionable)

Method

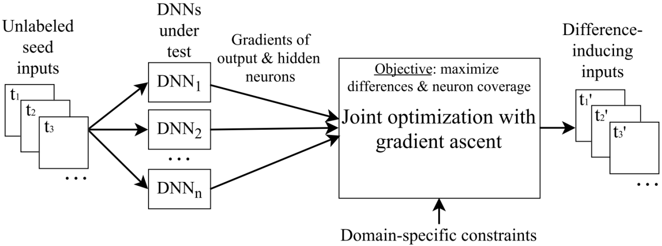


Figure 4: DeepXplore workflow.

- Maximizing neuron coverage: Generate test case that can activate more neurons.
- Oracle: Test multiple target DNN together and find those test cases that is supported by all DNNs except one. (Actually, in the experiment there's always 3 target models, so it's always 2-1)

Doing two things together.

DeepXplore Algorithm

Algorithm 1 Test input generation via joint optimization

Input: **seed_set** ← unlabeled inputs as the seeds
dnns ← multiple DNNs under test
λ_1 ← parameter to balance output differences of DNNs (Equation 2)
λ_2 ← parameter to balance coverage and differential behavior
s ← step size in gradient ascent
t ← threshold for determining if a neuron is activated
p ← desired neuron coverage
cov_tracker ← tracks which neurons have been activated

```
/* main procedure */
gen_test := empty set
for cycle(x ∈ seed_set) do  // infinitely cycling through seed_set
    /* all dnns should classify the seed input to the same class */
    c = dnns[0].predict(x)
    d = randomly select one dnn from dnns
    while True do
        obj1 = COMPUTE_OBJ1(x, d, c, dnns, λ1)
        obj2 = COMPUTE_OBJ2(x, dnns, cov_tracker)
        obj = obj1 + λ2 · obj2
        grad = ∂obj / ∂x
        /*apply domain-specific constraints to gradient*/
        grad = DOMAIN_CONSTRNTS(grad)
        x = x + s · grad  //gradient ascent
        if d.predict(x) ≠ (dnns-d).predict(x) then
            /* dnns predict x differently */
            gen_test.add(x)
            Update cov_tracker
            break
    if DESIRED_COVERAGE_ACHVD(cov_tracker) then
        return gen_test
/* utility functions for computing obj1 and obj2 */
procedure COMPUTE_OBJ1(x, d, c, dnns, λ1)
    rest = dnns - d
    loss1 := 0
    for dnn in rest do
        loss1 += dnn_c(x)  //confidence score of x being in class c
    loss2 := d_c(x)  //d's output confidence score of x being in class c
    return (loss1 - λ1·loss2)
procedure COMPUTE_OBJ2(x, dnns, cov_tracker)
    loss := 0
    for dnn ∈ dnns do
        select a neuron n inactivated so far using cov_tracker
        loss += n(x)  //the neuron n's output when x is the dnn's input
    return loss
```

This algorithm does an optimization in two objectives together.

DeepXplore Algorithm(Contd.)

Objective 1: Find a test case that maximize the difference for one model and the other models. Formula:

$$\text{obj}_1(x) = \Sigma_{i \neq j} F_i(x)[c] - \lambda_1 * F_j(x)[c]$$

For example, Suppose three classifiers have the following result on input x:

	Model A	Model B	Model C
Car	0.7	0.8	0.9
Truck	0.2	0.1	0.1
Bike	0.1	0.1	0

The algorithm pick Model C as the target, that is, try to generate a test case x' based on x that make Model C generate a different answer.
Suppose $\lambda_1 = 2$
Then the output of object 1 is $0.7 + 0.8 - 0.9 \times 2 = -0.3$

DeepXplore Algorithm(Contd.)

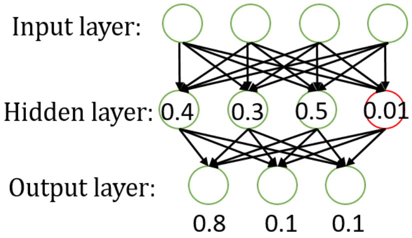
Objective 2: To find a test case that increase the neuron coverage:

$$\text{obj}_2 = out(n_i) \text{ that } out(n_i) > t$$

n_i is a randomly picked neuron. According to the algorithm, it's randomly found every time, each at a time. So it could be different neuron at steps of optimization.
Each time of the optimization select one neuron on every DNN and add them together.

DeepXplore Algorithm(Contd.)

Example:



In the case, the red neuron is not activated (threshold $= 0.1$) Therefore $\text{obj}_2 = 0.1$

DeepXplore Algorithm(Contd.)

1. Pick a test sample x (unlabeled). In the case, the classification result is 0 (car).
2. Randomly pick a model $d \in DNNs$. In the case, it's Model A.
3. Loop
- 3.1 Calculate $obj_1 = -0.3$
- 3.2 Calculate $obj_2 = 0.1$
- 3.3 $obj_{total} = Obj_1 + \lambda_2 obj_2$
- 3.4 Get the gradient $grad = \frac{\partial obj_{total}}{\partial x}$
- 3.5 $grad = \text{DOMAIN\_CONSTRAINTS}(grad)$
- 3.6 Let $x = x + \alpha * grad$
- 3.7 If d classifies differently (and other models still classifies the same), add x into the test set. Otherwise back to 3.1

4. If current set of test samples is good enough (on neuron coverage), stops. Otherwise go to 1.

Domain Constraints

In order to make the test cases generated meaningful:
1. Only make the image darker or brighter.
2. Change a rectangle part of the graph into random noise.
3. Only apply the negative perturbation on a small rectangle set of the model

Experiment Design

DeepXplore: Automated White-box Testing of Deep Learning Systems

Goal: 1. Show neuron coverage is a good metric

- Code coverage doesn't works
- Traditional inputs have small neuron coverage.
- Different class inputs activate different neurons

2. Show DeepXplore is a good method.

- Neuron coverage is good
- Execution time is acceptible

3. DeepXplore testing cases can be used to improve the model.

Experiment result - Neuron Coverage

Table 4: Comparison of code coverage and neuron coverage for 10 randomly selected inputs from the original test set of each DNN.

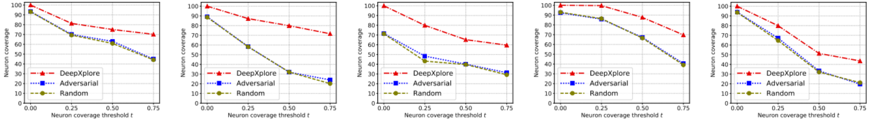
Dataset	Code Coverage			Neuron Coverage		
	C1	*C2*	*C3*	*C1*	*C2*	*C3*
MNIST	100%	100%	100%	32.7%	33.1%	25.7%
ImageNet	100%	100%	100%	1.5%	1.1%	0.3%
Driving	100%	100%	100%	2.5%	3.1%	3.9%
Contagio	100%	100%	100%	19.8%	17.3%	17.3%
Drebin	100%	100%	100%	16.8%	10%	28.6%

Table 5: Average number of overlaps among activated neurons for a pair of inputs of the same class and different classes. Inputs of different classes tend to activate different neurons.

	Total neurons	***Avg. no. of activated neurons***	***Avg. overlap***
Diff. class	268	83.6	45.9
Same class	268	84.1	74.2

- Code coverage is always 100%, while neuron coverage varies.
- For the same class, the number of activated neurons are much more similar

Experiment result - DeepXplore

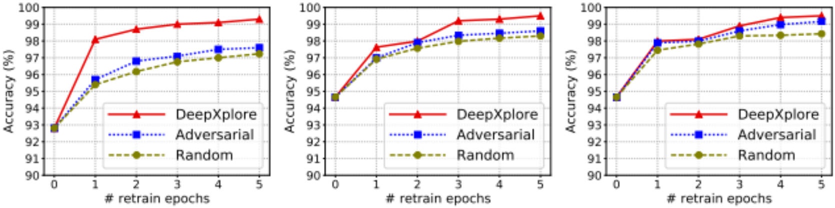


(a) MNIST (b) ImageNet (c) Driving (d) Contagio (e) Drebin

Figure 8: The neuron coverage achieved by the same number of inputs (1% of the original test set) produced by DeepXplore, adversarial testing [18], and random selection from the original test set. The plots show the change in neuron coverage for all three methods as the threshold t (defined in Section 5) increases. DeepXplore, on average, covers 34.4% and 33.2% more neurons than random testing and adversarial testing.

- DeepXplore achieves higher neuron coverage than baseline method
- However:
  1. Baseline method is not suitable for the task
  2. Only have neuron coverage result
- Result show DeepXplore achieves 100% neuron coverage in a short period of time, however they doesn't mention the threshold number here.

Experiment result - Usage of test input



(a) LeNet-1 **(b)** LeNet-4 **(c)** LeNet-5

Figure 9: Improvement in accuracy of three LeNet DNNs when the training set is augmented with the same number of inputs generated by random selection (“random”), adversarial testing (“adversarial”) [18], and DeepXplore.

- Use generate test suite to train.
  Problem: Result not very convincing.
- Detecting training data pollution attack: Highly doubtful