자동화된 뉴런 선택을 통한 심층 신경망 검사 기법

정보대학 컴퓨터학과 이석현
정보대학 컴퓨터학과 이다인
지도교수: 오학주

Introduction

심층 신경망은 게임, 그래픽과 같이 CS 내의 분야부터 의학 진단, 무인자동차와 같이 인간의 안전에 직접적인 영향을 줄 수 있는 분야까지 다양한 곳에 쓰이고 있다. 심층 신경망은 복잡한 구조와 블랙박스(Black-box)적인 특성 때문에 이를 효율적으로 검사하기 쉽지 않다.

소프트웨어를 테스트하는 대표적인 기법은 입력값을 생성하며 오류가 발생하는 상황을 찾는 퍼징(Fuzzing)인데, 신경망에 적용하는 상황에서는 오류가 발생하는 뉴런을 선택하여 그 뉴런의 입력에 대한 기울기를 사용하는 방식을 택한다. 본 연구에서는 뉴런의 특징을 벡터화하여, 상황에 맞게 뉴런을 선택하는 전략을 학습을 통해 생성하여 더욱 효율적으로 심층 신경망을 검사하고자 한다. 이 때, 효율성은 현존 기법들보다 1) 정의한 커버리지(Coverage)에 대해서 더 높은 수치를 달성하고, 2) 더 다양하고 많은 오류를 찾는 것을 의미한다.

Approach

본 연구에서는 심층 신경망을 검사하기 위해 기존의 화이트박스 심층 신경망 검사에 두 가지 기법을 추가하였다. 제안하는 알고리즘은 아래와 같다.

A. 벡터화된 뉴런 선택 전략

뉴런의 특징을 표현하기 위해서 아래의 표와 같이 총 29개의 피처(Feature)를 디자인하였다. 29개의 피처는 17개의 정적 피처, 12개의 동적 피처로 구성되어 있다. 세부적으로 17개의 정적 피처 중 11개는 신경망의 층과 관련되어 있고, 6개는 뉴런과 관련되어 있으며, 12개의 동적 피처는 모두 뉴런과 관련되어 있다. 뉴런(n)의 피처 벡터(F_n)와 같은 차원의 실수 벡터로 표현되는 선택 전략(p)을 이용하여 아래와 같이 정해진 개수(k)의 뉴런을 선택한다.

$$F_n = \langle F_{n,1}, F_{n,2}, \cdots, F_{n,29} \rangle \, (F_{n,i} \in \{0,1\})$$

$$p \in \mathbb{R}^{29}$$

$$score_n = F_n \cdot p$$

$$Strategy_p(N) = \left\{ n \,|\, n \in \underset{S \subseteq N \wedge |S| = k}{\arg\max} \left(\sum_{n \in S} score_n \right) \right\}$$

B. 학습을 이용한 뉴런 선택 전략 생성

각 상황에 맞게 효율적인 선택 전략을 생성하기 위해서, 학습 과정 중에 생성된 정보(D)를 바탕으로 새로운 선택 전략의 집합(P)을 생성하는 방법은 아래와 같다.

$$SD^* = \underset{SD \subseteq D}{\arg\max} \left| \bigcup_{(p, C_p) \in SD} Cp \right|$$

$$\mu = \frac{\sum_{(p, C_p) \in SD^*} p}{|SD^*|}, \Sigma = \frac{\sum_{(p, C_p) \in SD^*} (p-\mu)(p-\mu)^T}{|SD^*| - 1}$$

$$P = \{ p_1, p_2, \cdots, p_\eta \,|\, p_i \sim \aleph(\mu, \Sigma) \}$$

#	Description
1	Neuron located in front 25% layers
2	Neuron located in front 25-50% layers
3	Neuron located in front 50-75% layers
4	Neuron located in front 75-100% layers
5	Neuron in a normalization layer
6	Neuron in a pooling layer
7	Neuron in a convolution layer
8	Neuron in a dense layer
9	Neuron in an activation layer
10	Neuron in a layer with multiple input source
11	Neuron that does not belong to of 5-10 features
12	Neuron with top 10% weights
13	Neuron with weights between top 10% and 20%
14	Neuron with weights between top 20% and 30%
15	Neuron with weights between top 30% and 40%
16	Neuron with weights between top 40% and 50%
17	Neuron with weights in bottom 50%
18	Neuron activated when an adversarial input is found
19	Neuron never activated
20	Neuron with the number of activations (top 10%)
21	Neuron with activation numbers (top 10-20%)
22	Neuron with activation numbers (top 20-30%)
23	Neuron with activation numbers (top 30-40%)
24	Neuron with activation numbers (top 40-50%)
25	Neuron with activation numbers (top 50-60%)
26	Neuron with activation numbers (top 60-70%)
27	Neuron with activation numbers (top 70-80%)
28	Neuron with activation numbers (top 80-90%)
29	Neuron with activation numbers (top 90-100%)

Algorithm 1 Our Approach for Testing Neural Networks

```
procedure ADAPT(N, I, Cov)
    C ← Cov(Forward(N, I))
    P ← {p_1, ··· , p_η | p_i ∼ U([−1, 1])^29}
    D ← ∅
    repeat
        for all p ∈ P do
            C_p ← ∅
            W ← {I}
            while W ≠ ∅ do                          ← 벡터화된 뉴런 선택 전략
                I' ← Pick an input from W
                W ← W \ {I'}
                V ← Strategy_p(Forward(N, I'))
                for n = 1 to η_t do
                    I' ← I' + λ · ∂(Σ_{v∈V} v)/∂I'
                    O' ← Forward(N, I')
                    if Cov(O') ⊈ C ∧ Obj(I, I') then
                        W ← W ∪ {I'}
                C ← C ∪ Cov(O')                     ← 생성된 정보 저장
                C_p ← C_p ∪ Cov(O')
            D ← D ∪ {(p, C_p)}
        P ← Learning(D)                             ← 학습을 이용한 전략 생성
    until testing budget expires (e.g. timeout)
    return |C|
```

Conclusion

효율적으로 심층 신경망을 검사하기 위해 선택 전략을 벡터화하는 것과 검사 과정 중 생성된 정보를 바탕으로 뉴런 선택 전략을 생성하는 방법을 제안하였다. 다양한 실험 환경에서 다른 툴들과 비교했을 때 지속적으로 높은 커버리지를 달성하였고, 생성하는 이미지 또한 다양하다.

Experiments

Figure 1. Effectiveness for increasing NC

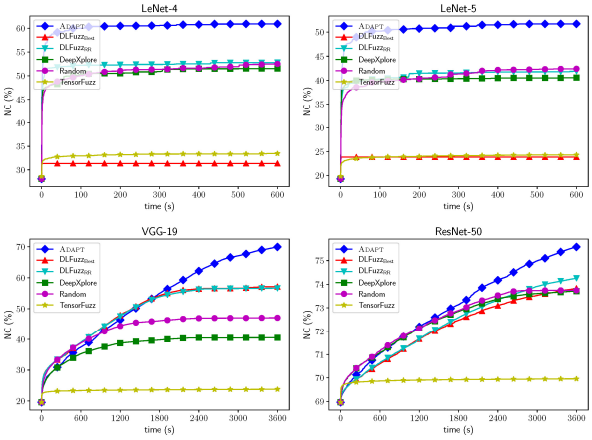

Figure 2. Effectiveness for increasing TKNC

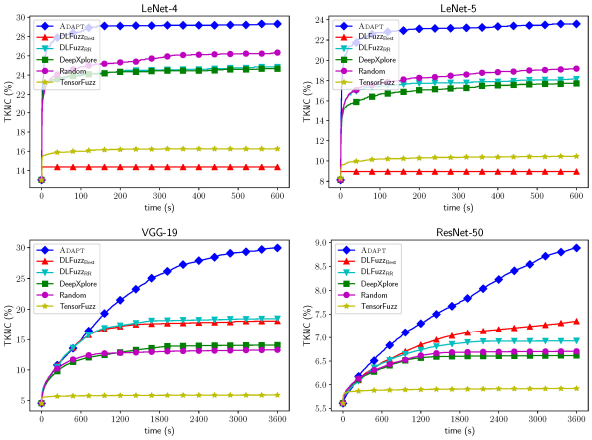

Table 1. Adversarial inputs found with NC

Model	Technique	# of Mutations	# of Adv. Inputs	# of Labels	# of Seeds
LeNet-4	ADAPT	34012.0	246.2	**3.7**	**20/20**
	TensorFuzz	98796.8	0.0	0.0	0/20
	Random	39769.3	221.4	2.0	15/20
	DeepXplore	39917.4	20.4	1.2	12/20
	$DLFuzz_{Best}$	36866.8	**635.1**	0.2	3/20
	$DLFuzz_{RR}$	37211.0	62.2	1.6	17/20
LeNet-5	ADAPT	32950.1	**587.4**	**5.2**	**20/20**
	TensorFuzz	90508.6	0.0	0.0	0/20
	Random	36637.5	268.4	2.4	17/20
	DeepXplore	37178.2	36.1	1.6	16/20
	$DLFuzz_{Best}$	37335.2	2.0	0.2	4/20
	$DLFuzz_{RR}$	36400.6	193.0	2.0	17/20
VGG-19	ADAPT	12303.9	**4566.5**	**27.0**	**10/10**
	TensorFuzz	15583.4	186.8	0.1	1/10
	Random	13038.3	1222.3	9.4	7/10
	DeepXplore	13179.3	1575.4	7.3	8/10
	$DLFuzz_{Best}$	12883.0	2581.8	15.8	**10/10**
	$DLFuzz_{RR}$	12790.8	2686.9	15.8	9/10
ResNet-50	ADAPT	8461.6	3982.5	3.0	**7/10**
	TensorFuzz	12279.6	1948.0	0.3	2/10
	Random	9422.7	3005.0	3.0	**7/10**
	DeepXplore	9221.0	3043.0	2.6	**7/10**
	$DLFuzz_{Best}$	8914.5	3655.1	3.1	**7/10**
	$DLFuzz_{RR}$	9389.4	**4083.2**	**3.6**	**7/10**

Table 2. Adversarial inputs found with TKNC

Model	Technique	# of Mutations	# of Adv. Inputs	# of Labels	# of Seeds
LeNet-4	ADAPT	33041.0	253.0	**2.8**	**18/20**
	TensorFuzz	96388.4	0.0	0.0	0/20
	Random	38752.4	204.4	1.8	13/20
	DeepXplore	38618.4	13.8	1.0	9/20
	$DLFuzz_{Best}$	36349.5	**1196.9**	0.0	1/20
	$DLFuzz_{RR}$	38111.2	47.4	1.1	10/20
LeNet-5	ADAPT	30927.2	**531.6**	**4.4**	**19/20**
	TensorFuzz	91278.2	0.0	0.0	0/20
	Random	36742.6	244.8	2.0	15/20
	DeepXplore	37295.2	105.8	1.1	10/20
	$DLFuzz_{Best}$	34338.7	0.2	0.1	2/20
	$DLFuzz_{RR}$	35910.2	193.8	1.4	15/20
VGG-19	ADAPT	12138.9	**3155.8**	**32.0**	**10/10**
	TensorFuzz	15130.9	181.2	0.1	1/10
	Random	13180.2	303.9	3.0	4/10
	DeepXplore	13053.9	619.3	9.1	5/10
	$DLFuzz_{Best}$	12710.7	891.1	8.3	7/10
	$DLFuzz_{RR}$	13022.7	1089.6	10.5	**10/10**
ResNet-50	ADAPT	8176.2	**3162.5**	**6.7**	**8/10**
	TensorFuzz	11779.4	1856.8	0.3	2/10
	Random	9215.4	1625.5	1.7	7/10
	DeepXplore	8944.9	1529.3	2.0	7/10
	$DLFuzz_{Best}$	9207.9	2381.0	3.0	7/10
	$DLFuzz_{RR}$	9131.2	1937.9	2.8	7/10