졸업프로젝트 보고서

자동화된 뉴런 선택을 통한 심층 신경망 검사 기법

지도교수 : 오학주

2019년 12월 06일

고려대학교 정보대학 컴퓨터학과

이석현 (2016320125)
이다인 (2016320202)

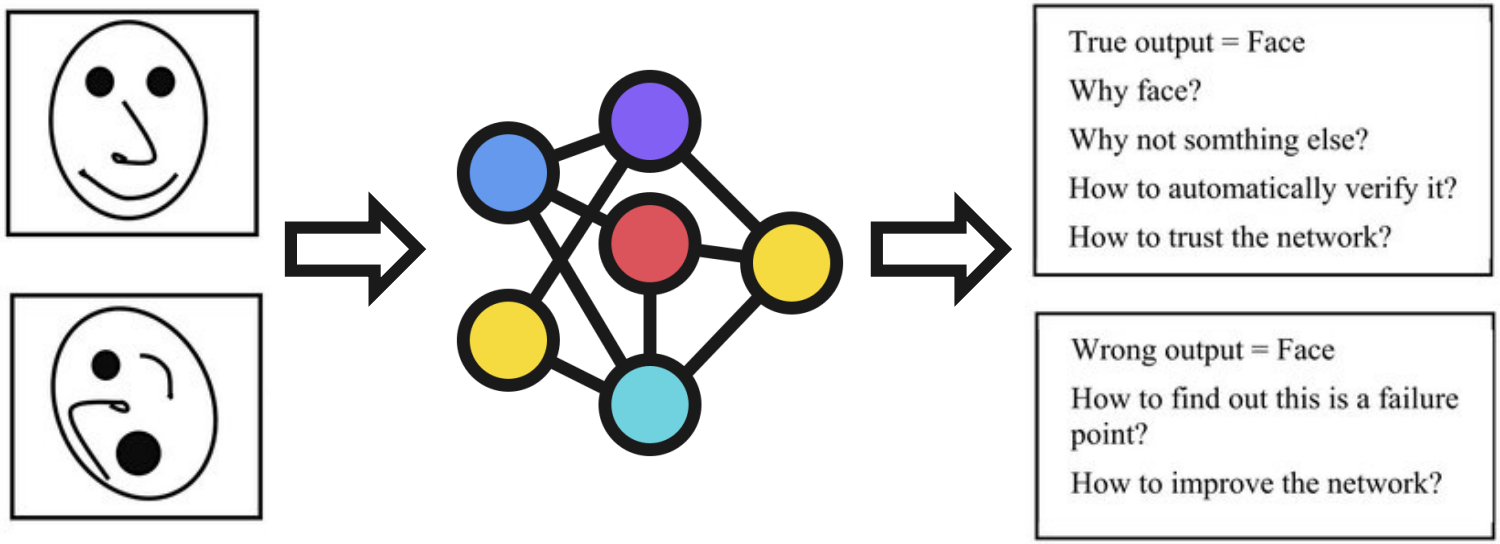
Contents

- ❑ Introduction
- ❑ Related Work (Project)
- ❑ Solution Approach (Main Idea)
- ❑ Experiment / 성능평가
- ❑ Conclusion

1. Introduction

❑ Problem Statement

- ❍ 심층 신경망은 높은 복잡도와 내부의 값의 의미를 알기 어렵기때문에 이를 검사하기 어렵다.
- ❍ 현존하는 검사 기법은 1) 계산 복잡도가 지나치게 높거나, 2) 연구자가 디자인한 효율적이지 않은 뉴런 선택 전략을 이용한다.



1. Introduction

❑ Pains and Needs

- ❍ 계산 복잡도가 낮은 검사 기법이 개발될 경우,

1) 실제로 사용되는 큰 신경망을 검사 할 수 있다.

- ❍ 뉴런 선택 전략을 자동으로 생성해주는 알고리즘이 개발될 경우,

1) 연구자가 새로운 알고리즘을 생성하기 위한 노력을 줄일 수 있고,

2) 다양한 실험 상황 및 신경망에서 보다 효율적으로 동작할 수 있다.

❑ Importance

- ❍ 기계학습, 인공지능 관련 학계와 실제 모델을 적용하는 산업에서도 더욱 쉽고 효율적으로 심층 신경망을 검사할 수 있다.

2. Related Works (Project)

❑ 내부 값을 이용하여 입력을 생성하는 심층 신경망 검사 (White-box)

- ❍ 연구자가 디자인한 간단한 뉴런 선택 전략에 따라 동작 (DeepXplore, DLFuzz)
- ❍ 계산 복잡도가 지나치게 높아 매우 작은 신경망만 검사할 수 있음 (DeepConcolic)

❑ 내부 값을 참고하여 입력을 생성하는 심층 신경망 검사 (Grey-box)

- ❍ 연구자가 입력을 변환할 수 있는 수 많은 방법을 직접 디자인해야함 (DeepTest, DeepHunter)
- ❍ 생성된 입력이 원본과 차이가 큼 (DeepTest, DeepHunter, Tensorfuzz)
- ❍ 입력을 원하는 방향으로 만들지 못해 효율적이 못함 (DeepTest, DeepHunter, Tensorfuzz)

2. Related Works (Project)

❑ 심층 신경망에 대한 커버리지

- ❍ 심층 신경망의 구조를 이용하여 신경망을 얼마나 탐색해봤는지 수치화 (Neuron Coverage, Top-k Neuron Coverage)
- ❍ 심층 신경망의 학습 과정 중에서 얻을 수 있는 값을 바탕으로 신경망을 얼마나 탐색해봤는지 수치화 (K-multisection Neuron Coverage, Surprise Adequacy)

❑ 학습을 이용한 소프트웨어 검사

- ❍ 탐색 전략을 벡터로 표현하여 학습을 이용하여 탐색 전략을 생성해서 기존의 소프트웨어를 검사하는 방법 (ParaDySE, Chameleon)

❑심층 신경망 적대적 공격

- ❍ 신경망이 잘못 동작하도록 하게 하는 입력을 생성하는 연구로 신경망의 다양한 상태를 검사하고자 하는 본 연구와는 다른 방향의 연구

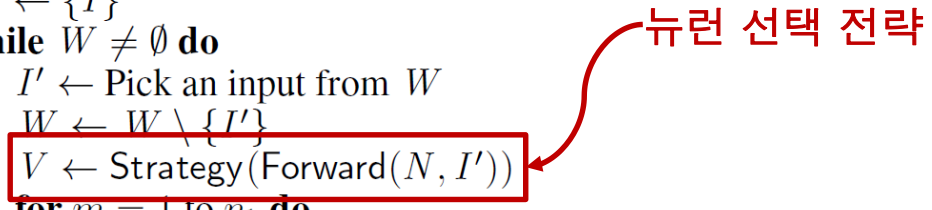
3. Solution Approach (Main Idea)

❑ 벡터화된 뉴런 선택 전략

- ❍ 내부 값을 이용하여 입력을 생성하는 심층 신경망 검사 기법 중 뉴런을 선택하는 전략을 벡터화 함

Algorithm 1 White-box Testing for Neural Networks

```
1:  procedure TESTINGDNN(N, I, Cov)
2:      C ← Cov(Forward(N, I))
3:      repeat
4:          W ← {I}
5:          while W ≠ ∅ do
6:              I' ← Pick an input from W
7:              W ← W \ {I'}
8:              V ← Strategy(Forward(N, I'))
9:              for m = 1 to η1 do
10:                 I' ← I' + λ · ∂(Σ_{v∈V} v)/∂I'
11:                 O' ← Forward(N, I')
12:                 if Cov(O') ⊈ C ∧ Obj(I, I') then
13:                     W ← W ∪ {I'}
14:                     C ← C ∪ Cov(O')
15:     until testing budget expires (e.g. timeout)
16:     return |C|
```



3. Solution Approach (Main Idea)

❑ 벡터화된 뉴런 선택 전략

- 총 17개의 정적 피처, 총 12개의 동적 피처를 디자인
- 세부적으로 17개의 정적 피처 중, 11개의 피처는 신경망의 층과 관련되어 있고, 6개는 뉴런과 관련되어 있음
- 12개의 동적 피처는 모두 뉴런과 관련되어 있음

#	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%)

3. Solution Approach (Main Idea)

❑ 벡터화된 뉴런 선택 전략

- 뉴런(n)의 피처 벡터와

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

- 벡터화된 뉴런 선택 전략을 이용하여

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

- 각 뉴런의 점수을 계산 한 후,

$$score_n = F_n \cdot p$$

- 점수가 가장 높은 k개의 뉴런을 선택

3. Solution Approach (Main Idea)

❑ 학습을 이용한 선택 전략 생성

Algorithm 2 Our Approach for Testing Neural Networks

```
1: procedure ADAPT(N, I, Cov)
2:     C ← Cov(Forward(N, I))
3:     P ← {p_1, ··· , p_η2 | p_i ∼ U([−1, 1])^29}
4:     D ← ∅
5:     repeat
6:         for all p ∈ P do
7:             C_p ← ∅
8:             W ← {I}
9:             while W ≠ ∅ do
10:                I′ ← Pick an input from W
11:                W ← W \ {I′}
12:                V ← Strategy_p(Forward(N, I′))
13:                for m = 1 to η1 do
14:                    I′ ← I′ + λ · ∂(Σ_{v∈V} v)/∂I′
15:                    O′ ← Forward(N, I′)
16:                    if Cov(O′) ⊈ C ∧ Obj(I, I′) then
17:                        W ← W ∪ {I′}
18:                        C ← C ∪ Cov(O′)
19:                    C_p ← C_p ∪ Cov(O′)
20:            D ← D ∪ {(p, C_p)}
21:        P ← Learning(D)
22:    until testing budget expires (e.g. timeout)
23:    return |C|
```

벡터화된 뉴런 선택 전략

검사 과정 중 생성된 정보

학습을 이용한 전략 생성

3. Solution Approach (Main Idea)

❑ 학습을 이용한 선택 전략 생성

- ❍ 커버리지의 합집합이 최대가 되도록 선택 전략을 선택하고,

$$SD = \underset{SD' \subseteq D}{\operatorname{argmax}} \left| \bigcup_{(p,C_p) \in SD'} C_p \right|$$

- ❍ 선택된 전략의 평균과 분산을 이용하여,

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

- ❍ 새로운 뉴런선택 전략들을 생성

$$\{p_1, p_2, \cdots, p_\eta | p_i \sim \mathcal{N}(\mu, \Sigma)\}$$

4. Experiments / 성능 평가

❑ 실험에 이용한 심층 신경망과 데이터셋

Dataset	Model	# of Neurons	# of Layers	Accuracy
MNIST	LeNet-4	148	9	0.985
	LeNet-5	268	10	0.988
ImageNet	VGG-19	16,168	26	0.713
	ResNet-50	94,123	177	0.764

❑ 실험에 이용한 커버리지 평가 방법

- Neuron Coverage (SOSP'17)
- Top-k Neuron Coverage (ASE'18)

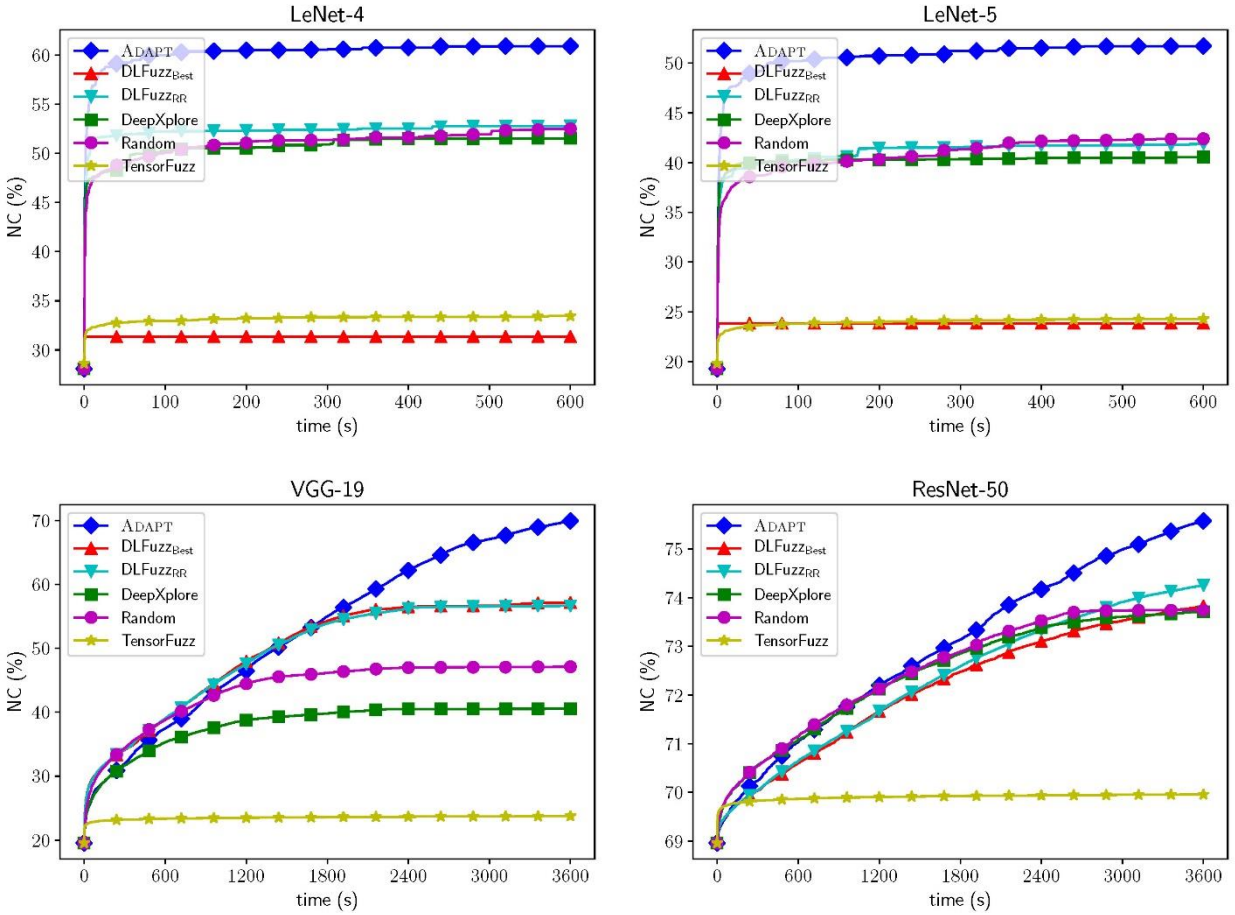
4. Experiments / 성능 평가

❑ 실험에 이용한 툴

- Adapt: 제안하는 툴
- $\text{DLFuzz}_{\text{RR}}$(FSE'19): 논문에서 제안한 3가지 선택 전략의 라운드로빈
  1) 가장 많이 탐색해본 뉴런을 선택
  2) 가장 적게 탐색해본 뉴런을 선택
  3) 가장 가중치가 큰 뉴런을 선택
- $\text{DLFuzz}_{\text{Best}}$(FSE'19): 논문에서 제안한 가장 좋은 선택 전략 (1번 전략)
- DeepXplore(SOSP'17): 한번도 가보지 않은 뉴런을 선택
- Random: 전체 뉴런 중에서 임의로 선택
- Tensorfuzz(arXiv'18): 심층 신경망을 이용하여 기울기를 구하지 않고 커버리지만 이용하여 다음 입력을 생성하는 툴

4. Experiments / 성능 평가

❑ Neuron Coverage



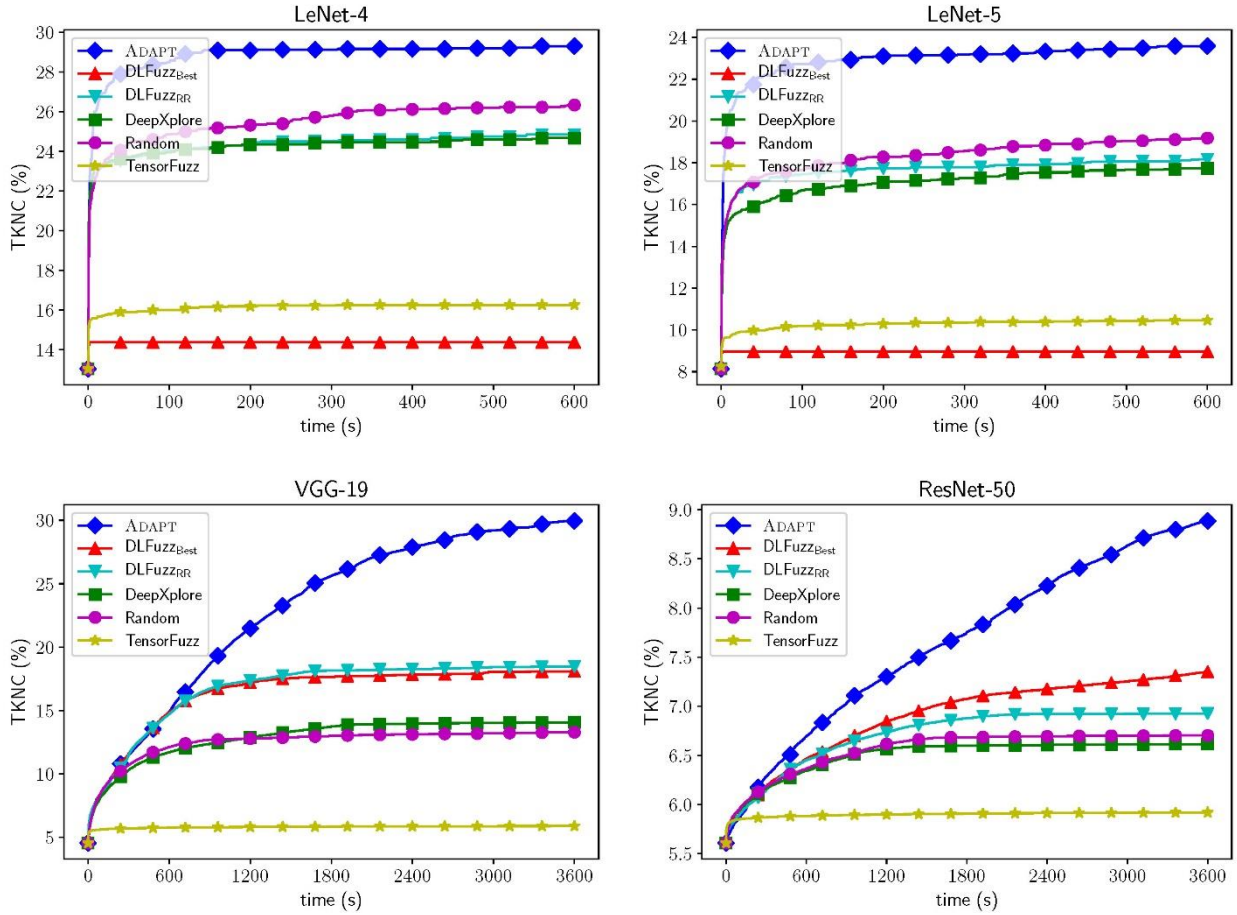
4. Experiments / 성능 평가

❑ Neuron Coverage

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
	$\text{DLFuzz}_{\text{Best}}$	36866.8	**635.1**	0.2	3/20
	$\text{DLFuzz}_{\text{RR}}$	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
	$\text{DLFuzz}_{\text{Best}}$	37335.2	2.0	0.2	4/20
	$\text{DLFuzz}_{\text{RR}}$	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
	$\text{DLFuzz}_{\text{Best}}$	12883.0	2581.8	15.8	**10/10**
	$\text{DLFuzz}_{\text{RR}}$	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**
	$\text{DLFuzz}_{\text{Best}}$	8914.5	3655.1	3.1	**7/10**
	$\text{DLFuzz}_{\text{RR}}$	9389.4	**4083.2**	**3.6**	**7/10**

4. Experiments / 성능 평가

❑ Top-k Neuron Coverage



4. Experiments / 성능 평가

❑ Top-k Neuron Coverage

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
	$\text{DLFuzz}_{\text{Best}}$	36349.5	**1196.9**	0.0	1/20
	$\text{DLFuzz}_{\text{RR}}$	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
	$\text{DLFuzz}_{\text{Best}}$	34338.7	0.2	0.1	2/20
	$\text{DLFuzz}_{\text{RR}}$	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
	$\text{DLFuzz}_{\text{Best}}$	12710.7	891.1	8.3	7/10
	$\text{DLFuzz}_{\text{RR}}$	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
	$\text{DLFuzz}_{\text{Best}}$	9207.9	2381.0	3.0	7/10
	$\text{DLFuzz}_{\text{RR}}$	9131.2	1937.9	2.8	7/10

5. Conclusion

❑ Contributions & Summary

- ❍ 심층 신경망을 효율적으로 검사하기 위해서 1) 벡터화된 선택 전략을 디자인하고 2) 학습을 이용하여 적절한 선택 전략을 생성하는 알고리즘을 제안하였다.
- ❍ 기존에 제안된 툴과 비교했을 때 모든 실험에서 지속적으로 가장 높은 커버리지를 달성하였고, 생성하는 이미지도 가장 다양하다.

❑ Future work

- ❍ 사진 분류를 위한 심층 신경망을 넘어 다른 종류의 일을 처리하는 심층 신경망으로 확장하려 한다.
- ❍ 모든 모델에서 잘 동작 할 수 있도록 하는 선택 전략들을 미리 학습을 통해 구하려 한다.