Feature-Guided Black-Box Safety Testing of Deep Neural Networks

Youcheng Sun, Xiaowei Huang, and Daniel Kroening
Arxiv 2018

Table of Contents

Table of Contents

Introduction

Artificial intelligence systems are typically implemented in software.

However, (white-box) testing for traditional software cannot be directly applied to DNNs, because the software that implements DNNs does not have suitable structure.

In particular, DNNs do not have traditional flow of control and thus it is not obvious how to define criteria such as branch coverage for them.

In this paper, we bridge this gap by proposing a novel (white-box) testing methodology for DNNs, including both test coverage criteria and test case generation algorithms.

Introduction

Any approach to testing DNNs needs to consider the distinct features of DNNs, such as

- The syntactic connections between neurons in adjacent layers (neurons in a given layer interact with each other and then pass information to higher layers)
- The ReLU activation functions, and
- The semantic relationship between layers (e.g., neurons in deeper layers represent more complex features)

Introduction

The contributions of this paper are three-fold.

First, they propose four test criteria, inspired by the MC/DC test criteria from traditional software testing, that fit the distinct features of DNNs.

There exist two coverage criteria for DNNs: neuron coverage and safety coverage, both of which have been proposed recently.

Neuron coverage is too coarse: 100% coverage can be achieved by a simple test suite comprised of few input vectors from the training dataset.

Safety coverage is black-box, too fine, and it is computationally too expensive to compute a test suite in reasonable time.

Their four proposed criteria are incomparable with each other, and complement each other in guiding the generation of test cases.

Introduction

Second, they develop an automatic test case generation algorithm for each of our criteria.

The algorithms produce a new test case by perturbing a given one using linear programming (LP).

LP can be solved efficiently in practice, and thus, their test case generation algorithms can generate a test suite with low computational cost.

Introduction

Finally, they implement our testing approaches in a software tool named *DeepCover* (available), and validate it by conducting experiments on a set of DNNs obtained by training on the MNIST dataset.

Table of Contents

Background

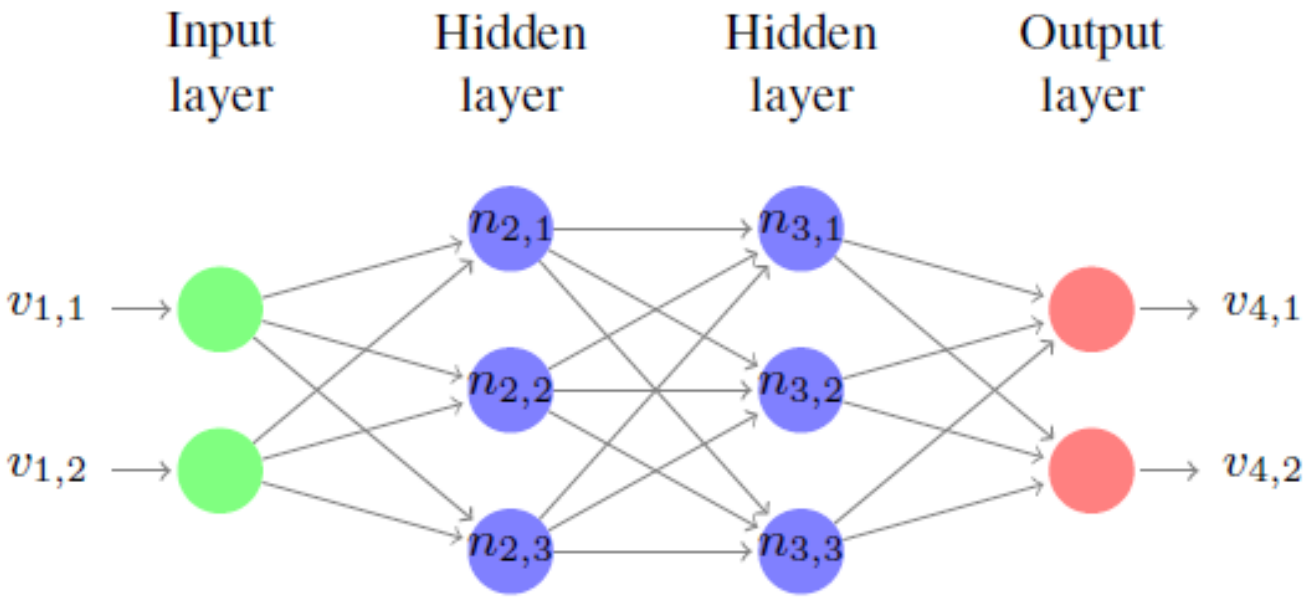


Figure: Given one particular input x, we say that the neural work N is instantiated and we use $N[x]$ to denote this instance of the network.

Background

$$v_{k,l} = \delta_{k,l} + \sum_{1 \le h \le s_{k-1}} w_{k-1,h,l} \cdot v_{k-1,h}$$

$$v_{k,l} = ReLU(v_{k,l}) = \begin{cases} v_{k,l} & \text{if } v_{k,l} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$label = \text{argmax}_{1 \le l \le s_K} \{v_{K,l}\}$$

$$sign(v_{k,l}[x]) = \begin{cases} +1 & \text{if } u_{k,l}[x] = v_{k,l}[x] \\ -1 & \text{otherwise} \end{cases}$$

Table of Contents

Adequacy Criteria for Testing Deep Neural Networks

Test Coverage and MC/DC

Let **N** be a set of neural networks, **R** the set of requirements, and **T** the set of test suites.

Definition 1 ([5]). *A test adequacy criterion, or a test coverage metric, is a function* $M : \mathsf{N} \times \mathsf{R} \times \mathsf{T} \to [0, 1]$.

Usually, the greater the number $M(N, R, T)$, the more adequate the testing.

Their new criteria for DNNs are inspired by established practices in software testing, in particular MC/DC test coverage, but are designed for the specific features of neural networks.

Adequacy Criteria for Testing Deep Neural Networks

Test Coverage and MC/DC

Modified Condition/Decision Coverage (MC/DC) is a method of ensuring adequate testing for safety-critical software.

At its core is the idea that if a choice can be made, all the possible factors (conditions) that contribute to that choice (decision) must be tested.

$$d \iff ((a > 3) \vee (b = 0)) \wedge (c \neq 4) \tag{5}$$

contains the three conditions $(a > 3)$, $(b = 0)$ and $(c \neq 4)$. The following six test cases provide 100% MC/DC coverage:

1. $(a > 3)$=true, $(b = 0)$=true, $(c \neq 4)$=true
2. $(a > 3)$=false, $(b = 0)$=false, $(c \neq 4)$=false
3. $(a > 3)$=false, $(b = 0)$=false, $(c \neq 4)$=true
4. $(a > 3)$=false, $(b = 0)$=true, $(c \neq 4)$=true
5. $(a > 3)$=false, $(b = 0)$=true, $(c \neq 4)$=false
6. $(a > 3)$=true, $(b = 0)$=false, $(c \neq 4)$=true

The first two test cases already satisfy both the *condition coverage* (i.e., all possibilities of the conditions are exploited) and the *decision coverage* (i.e., all possibilities of the decision d are exploited).

The last four cases are needed because for MC/DC each condition should evaluate to true and false at least once and should also affect the decision outcome.

The information represented by a neuron in the next layer can be seen as a summary (implemented by the layer function, the weights, and the bias) of the information in the current layer.

The core idea of our criteria is to ensure that not only the presence of a feature needs to be tested but also the effects of less complex features on a more complex feature must be tested.

Adequacy Criteria for Testing Deep Neural Networks

Decisions and Conditions in DNNs

Definition 2. *A neuron pair* $(n_{k,i}, n_{k+1,j})$ *are two neurons in adjacent layers* k *and* $k+1$ *such that* $1 \leq k \leq K-1$, $1 \leq i \leq s_k$, *and* $1 \leq j \leq s_{k+1}$. *Given a network* $\mathcal{N}$, *we write* $\mathcal{O}(\mathcal{N})$ *for the set of its neuron pairs.*

Definition 3 (Sign Change). *Given a neuron* $n_{k,l}$ *and two test cases* x_1 *and* x_2, *we say that the sign change of* $n_{k,l}$ *is exploited by* x_1 *and* x_2, *denoted as* $sc(n_{k,l}, x_1, x_2)$, *if* $sign(v_{k,l}[x_1]) \neq sign(v_{k,l}[x_2])$. *We write* $\neg sc(n_{k,l}, x_1, x_2)$ *when the condition is not satisfied.*

Definition 4 (Value Change). *Given a neuron* $n_{k,l}$ *and two test cases* x_1 *and* x_2, *we say that the value change of* $n_{k,l}$ *is exploited with respect to a value function* g *by* x_1 *and* x_2, *denoted as* $vc(g, n_{k,l}, x_1, x_2)$, *if* $g(u_{k,l}[x_1], u_{k,l}[x_2])$=*true and* $\neg sc(n_{k,l}, x_1, x_2)$. *Moreover, we write* $\neg vc(g, n_{k,l}, x_1, x_2)$ *when the condition is not satisfied.*

(absolute change, relative change)

Definition 5 (Distance Change). *Given the set of neurons* $P_k = \{n_{k,l} \mid 1 \leq l \leq s_k\}$ *layer* k *and two test cases* x_1 *and* x_2*, we say that the distance change of* P_k *is exploite with respect to a distance function* h *by* x_1 *and* x_2*, denoted as* $dc(h, k, x_1, x_2)$*, if*

- $h(u_k[x_1], u_k[x_2])$ = *true, and*
- *for all* $n_{k,l} \in P_k$, $sign(v_{k,l}[x_1]) = sign(v_{k,l}[x_2])$.

We write $\neg dc(h, k, x_1, x_2)$ *when any of the conditions is not satisfied.*

Adequacy Criteria for Testing Deep Neural Networks

Covering Methods

Definition 6 (Sign-Sign Cover, or SS Cover). *A neuron pair* $\alpha = (n_{k,i}, n_{k+1,j})$ *is S*
covered by two test cases x_1, x_2*, denoted as* $cov_{SS}(\alpha, x_1, x_2)$*, if the following condition*
are satisfied by the network instances $\mathcal{N}[x_1]$ *and* $\mathcal{N}[x_2]$*:*

- $sc(n_{k,i}, x_1, x_2)$;
- $\neg sc(n_{k,l}, x_1, x_2)$ *for all* $p_{k,l} \in P_k \setminus \{i\}$;
- $sc(n_{k+1,j}, x_1, x_2)$.

The SS Cover is designed to provide evidence that the change of a condition neuron $n_{k,l}$'s activation sign independently affects the sign of the decision neuron $n_{k+1,j}$ in the next layer.

Definition 7 (Distance-Sign Cover, or DS Cover). *Given a distance function h, a neuron pair $\alpha = (n_{k,i}, n_{k+1,j})$ is DS-covered by two test cases x_1, x_2, denoted as $cov^h_{DS}(\alpha, x_1, x_2)$, if the following conditions are satisfied by the network instances $\mathcal{N}[x_1]$ and $\mathcal{N}[x_2]$:*

- $dc(h, k, x_1, x_2)$, *and*
- $sc(n_{k+1,j}, x_1, x_2)$.

Intuitively, the first condition describes the distance change of neurons in layer k and the second condition requests the sign change of the neuron $n_{k+1,j}$.

Adequacy Criteria for Testing Deep Neural Networks

Covering Methods

They expect their criteria can provide guidance to the test case generation algorithms for discovering un-safe cases, by working with two adjacent layers, which are finer than the input-output relation.

They notice that the label change in the output layer is the direct result of the changes to the activation values in the penultimate layer.

Adequacy Criteria for Testing Deep Neural Networks

Covering Methods

Definition 8 (Sign-Value Cover, or SV Cover). *Given a value function g, a neuron pair $\alpha = (n_{k,i}, n_{k+1,j})$ is SV-covered by two test cases x_1, x_2, denoted as $cov^g_{SV}(\alpha, x_1, x_2)$, if the following conditions are satisfied by the network instances $\mathcal{N}[x_1]$ and $\mathcal{N}[x_2]$:*

- $sc(n_{k,i}, x_1, x_2)$;
- $\neg sc(n_{k,l}, x_1, x_2)$ *for all* $p_{k,l} \in P_k \setminus \{i\}$;
- $vc(g, n_{k+1,j}, x_1, x_2)$.

Intuitively, the SV Cover observes significant change of a decision neuron's value, by independently modifying one its condition neuron's sign.

Adequacy Criteria for Testing Deep Neural Networks

Covering Methods

Definition 9 (Distance-Value Cover, or DV Cover). *Given a distance function h and a value function g, a neuron pair $\alpha = (n_{k,i}, n_{k+1,j})$ is DV-covered by two test cases x_1, x_2, denoted as $cov_{DV}^{h,g}(\alpha, x_1, x_2)$, if the following conditions are satisfied by the network instances $\mathcal{N}[x_1]$ and $\mathcal{N}[x_2]$:*

- $dc(h, k, x_1, x_2)$;
- $vc(g, n_{k+1,j}, x_1, x_2)$.

Intuitively, a DV cover targets the scenario that there is no sign change for a neuron pair, but the decision neuron's value is changed significantly.

Definition 10 (Test Requirement). *Given a network $\mathcal{N}$ and a covering method $f \in F$, a test requirement $\mathcal{R}f$ is to find a test suite $\mathcal{TR}f$ such that it covers all neuron pairs in $\mathcal{O}(\mathcal{N})$ with respect to f, i.e.,*

$$\forall \alpha \in \mathcal{O}(\mathcal{N}) \exists x_1, x_2 \in \mathcal{TR}f : f(\alpha, x_1, x_2) \tag{6}$$

Due to the one-to-one correspondence between f and $\mathcal{TR}f$, we may write $\mathcal{TR}f$ as $\mathcal{T}f$.

$F = \{cov_{SS}, cov^{d}_{DS}, cov^{g}_{SV}, cov^{d,g}_{DV}\}$

Intuitively, a test requirement **Rf** asks that all neuron pairs are covered by at least two test cases in **Tf** with respect to the covering method f.

Definition 11 (Test Criterion). *Given a network $\mathcal{N}$ and a covering method $f \in F$, the test criterion $M\mathcal{R}f$ for a test suite $\mathcal{T}$ is as follows:*

$$M\mathcal{R}f(\mathcal{N}, \mathcal{R}f, \mathcal{T}) = \frac{|\{\alpha \in \mathcal{O}(\mathcal{N}) \mid \exists x_1, x_2 \in T : f(\alpha, x_1, x_2)\}|}{|\mathcal{O}(\mathcal{N})|} \tag{7}$$

Due to the one-to-one correspondence between $M\mathcal{R}f(\mathcal{N}, \mathcal{R}f, \mathcal{T})$ and f, we may write $M\mathcal{R}f(\mathcal{N}, \mathcal{R}f, \mathcal{T})$ as $Mf(\mathcal{N}, \mathcal{T})$.

Intuitively, it computes the percentage of the neuron pairs that are covered by test cases in T with respect to the covering method f.

Adequacy Criteria for Testing Deep Neural Networks

Test Requirements and Criteria

	sign change (Def. 3)	distance change (Def. 5)
sign change (Def. 3)	$Mcov_{SS}(\mathcal{N}, \mathcal{T})$	$Mcov^{d}_{DS}(\mathcal{N}, \mathcal{T})$
value change (Def. 4)	$Mcov^{g}_{SV}(\mathcal{N}, \mathcal{T})$	$Mcov^{d,g}_{DV}(\mathcal{N}, \mathcal{T})$

Table 2: A set of adequacy criteria for testing the DNNs, inspired by the MC/DC code coverage criteria. The columns are changes to the conditions (i.e., the activations of neurons in layer k), and the rows are changes to the decision (i.e., the activation of a neuron in layer $k + 1$)

Table of Contents

Automated Test Case Generation

In this paper, they consider approaches based on constraint solving - (Linear Programming).

	sign change (Def. 3)	distance change (Def. 5)
sign change (Def. 3)	$Mcov_{SS}(\mathcal{N}, \mathcal{T})$	$Mcov^{d}_{DS}(\mathcal{N}, \mathcal{T})$
value change (Def. 4)	$Mcov^{g}_{SV}(\mathcal{N}, \mathcal{T})$	$Mcov^{d,g}_{DV}(\mathcal{N}, \mathcal{T})$

Table 2: A set of adequacy criteria for testing the DNNs, inspired by the MC/DC code coverage criteria. The columns are changes to the conditions (i.e., the activations of neurons in layer k), and the rows are changes to the decision (i.e., the activation of a neuron in layer $k + 1$)

Automated Test Case Generation

The function $\hat{f}$ represented by a DNN is highly non-linear and cannot be encoded with linear programming (LP) in general.

In this paper, for the efficient generation of a test case $x\prime$, they consider (1) an LP-based approach by fixing the activation pattern ap[x] according to a given input x, and (2) encoding a prefix of the network, instead of the entire network, with respect to a given neuron pair.

The variables used in the LP model are distinguished in **bold**.

Given an input x, the input variable $\mathbf{x}$, whose value is to be synthesized with LP, is required to have the identical activation pattern as x, i.e., ap[x] = ap[$\mathbf{x}$].

$$\{\mathbf{u_{k,i}} \geq 0 \wedge \mathbf{v_{k,i}} = \mathbf{u_{k,i}} \mid ap[x]_{k,i} \geq 0, k \in [2, K), i \in [1..s_k]\}$$
$$\cup\{\mathbf{u_{k,i}} < 0 \wedge \mathbf{v_{k,i}} = 0 \mid ap[x]_{k,i} < 0, k \in [2, K), i \in [1..s_k]\}$$
$$\{\mathbf{u}_{k,i} = \sum_{1 \leq j \leq s_{k-1}} \{w_{k-1,j,i} \cdot \mathbf{v}_{k-1,j}\} + \delta_{k,i} \mid k \in [2, K), i \in [1..s_k]\}$$

Please note that the resulting LP model $C[x] = C_1[x] \cap C_2[x]$ represents *a symbolic set of inputs* that have the identical activation pattern as x.

Definition 13. *Let $\mathcal{C}[1 \ldots k']$ be a partial encoding of $\mathcal{C}$ up to layer k'. More precisely, we write $\mathcal{C}[1 \ldots k']$ if we substitute K in Expression (8) and (9) by k'.*

Definition 14. *Let $\mathcal{C}[(k,i)]$ be the same set of constraints as $\mathcal{C}$, except that the constraint $\mathcal{C}_1[x](k,i)$ is substituted by $\mathcal{C}_1[x](k,i)_{-1}$.*

Definition 15. *Let $\mathcal{C}[+g] = \mathcal{C} \cup \{g\}$ and $\mathcal{C}[+h] = \mathcal{C} \cup \{h\}$.*

Automated Test Case Generation

Operations on activation patterns

In this section, they discuss a safety requirement that is independent of the test criteria. This is to check *automatically* whether a given test case x is a bug.

Definition 16 (Safety Requirement). *Given a finite set X of correctly labeled inputs, the safety requirement is to ensure that for all inputs x' that are* close *to an input $x \in X$, we have $\hat{f}(x') \neq f(x)$.*

Algorithm 1 Test-Gen

INPUT: DNN $\mathcal{N}$, covering method f
OUTPUT: test suite $\mathcal{T}$, adversarial examples $\mathcal{T}'$

1: **for** each neuron pair $n_{k,i}, n_{k+1,j} \in \mathcal{O}(\mathcal{N})$ **do**
2: $x_1, x_2 = get\_input\_pair(f, n_{k,i}, n_{k+1,j})$
3: **if** x_1, x_2 are valid inputs **then**
4: $\mathcal{T} \leftarrow \mathcal{T} \cup \{x_1, x_2\}$
5: **if** $\mathcal{N}[x_1].label \neq \mathcal{N}[x_2].label$ and $close(x_1, x_2)$ **then**
6: $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{x_1, x_2\}$
7: **return** $\mathcal{T}, \mathcal{T}'$

Algorithm 2 *get_input_pair* with the first argument being cov_{SS}

INPUT: cov_{SS}, neuron pair $n_{k,i}, n_{k+1,j}$
OUTPUT: input pair x_1, x_2

1: **for** each $x_1 \in data\_set$ **do**
2: $x_2 = lp\_call(\mathcal{C}[x_1][1..k][\{(k,i),(k+1,j)\}], \min ||x_2 - x_1||_\infty)$
3: **if** x_2 is a valid input **then return** x_1, x_2
4: **return** _, _

Algorithm 3 *get_input_pair* with the first argument being cov^d_{DS}

INPUT: cov^d_{DS}, neuron $n_{k,i}$
OUTPUT: input pair t_1, t_2

1: **for** each $x_1 \in data\_set$ **do**
2: $x_2 = lp\_call(\mathcal{C}[x_1][1..k-1](\{(k,i),h\}), \min ||x_2 - x_1||_\infty)$
3: **if** x_2 is a valid input **then return** x_1, x_2
4: **return** _, _

Algorithm 4 *get_input_pair* with the first argument being cov^{g}_{SV}

INPUT: cov^{g}_{SV}, neuron pair $n_{k,i}, n_{k+1,j}$
OUTPUT: input pair x_1, x_2

1: **for** each $x_1 \in data\_set$ **do**
2: $x_2 = lp\_call(\mathcal{C}[x_1][1..k](\{(k,i),(k+1,j),g\}), \min ||x_2 - x_1||_\infty)$
3: **if** x_2 is a valid input **then return** x_1, x_2
4: **return** _, _

Algorithm 5 *get_input_pair* with the first argument being $cov^{d,g}_{DV}$

INPUT: $cov^{d,g}_{DV}$, neuron $p_{k,i}$
OUTPUT: input pair t_1, t_2

1: **for** each $x_1 \in data\_set$ **do**
2: $x_2 = lp\_call(\mathcal{C}[x_1][1..k](\{(k+1,j),h\}), \min ||x_2 - x_1||_\infty)$
3: **if** x_2 is a valid input **then return** x_1, x_2
4: **return** _, _

Table of Contents

Experiments

They use the well-known MNIST Handwritten Image Dataset to train a set of 10 fully connected DNNs to perform classification.

Each DNN has an input layer of 28 x 28 = 784 neurons and an output layer of 10 neurons.

The number of hidden layers for each DNN is randomly sampled from the set of $\{3, 4, 5\}$ and at each hidden layer, the number of neurons are uniformly selected from 20 to 100.

Every DNN is trained until an accuracy of at least 97.0% is reached on the MNIST validation data

Experiments

Hypothesis 1: Neuron Coverage is Easy

In particular, a test suite with high neuron coverage is not sufficient to increase confidence in the neural network in safety-critical domains.

To demonstrate this, for each DNN tested, they randomly pick 25 images from the MNIST test dataset.

For each selected image, to maximize the neuron coverage, if an input neuron is not activated (i.e., its activation value is equal to 0), we sample its value from [0, 0.1].

Then we measure the neuron coverage of the DNN by using the generated test suite of 25 images. As a result, for all 10 DNNs, we obtain almost 100% neuron coverage.

Simple experiment here demonstrates that it is straight-forward to obtain a trivial test suite that has high neuron coverage but does not provide any

Experiments

Hypothesis 2: Random Testing is Inefficient

For each DNN, we first choose an image from the MNIST test dataset, denoted by x.

Subsequently, we randomly sample 10^5 inputs in the region bounded by x +- 0.1, and we check whether an adversarial example exists for the original image x.

Using this process, we have obtained adversarial examples for only a single one of the 10 DNNs, for the other nine DNNs, we did not observe any adversarial examples among the 10^5 randomly generated images.

1- DNN Bug finding

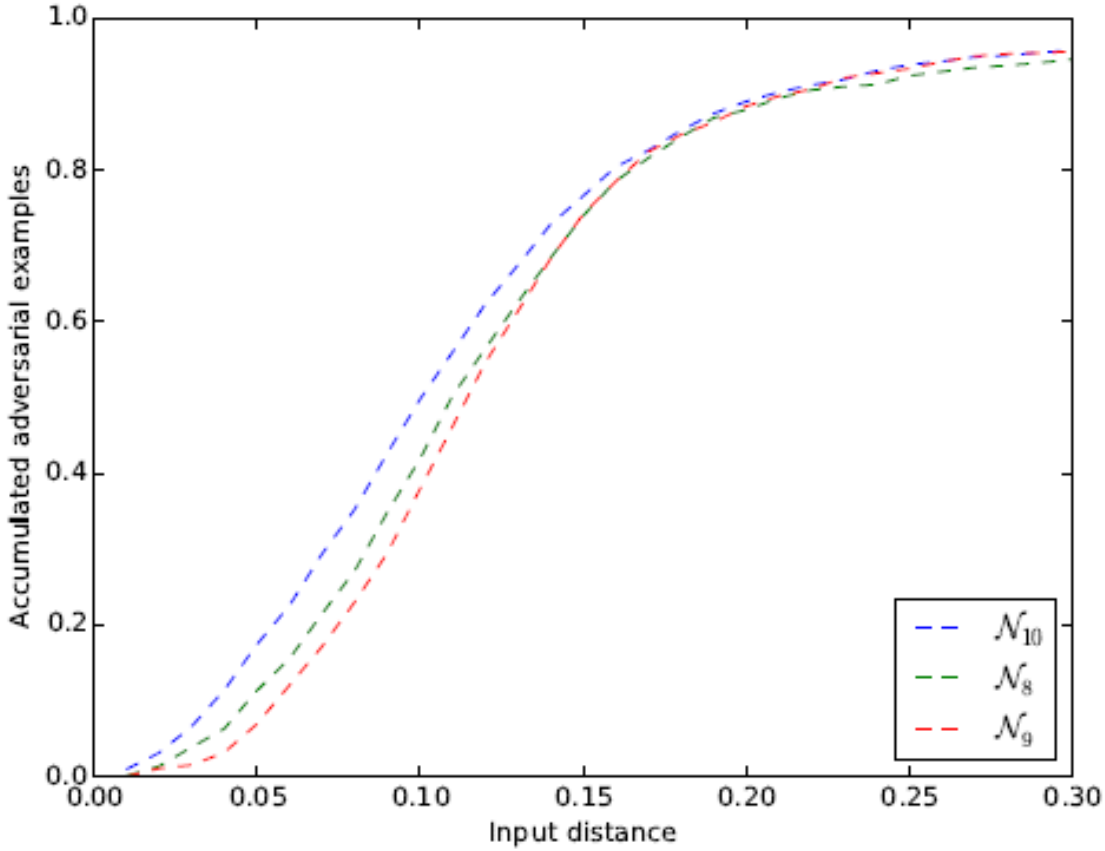
	hidden layers	$Mcov_{SS}$	$AEcov_{SS}$	$Mcov^{d}_{DS}$	$AEcov^{d}_{DS}$	$Mcov^{g}_{SV}$	$AEcov^{g}_{SV}$	$Mcov^{d,g}_{DV}$	$AEcov^{d,g}_{DV}$
$\mathcal{N}_1$	67x22x63	99.7%	18.9%	100%	15.8%	100%	6.7%	100%	21.1%
$\mathcal{N}_2$	59x94x56x45	98.5%	9.5%	100%	6.8%	99.9%	3.7%	100%	11.2%
$\mathcal{N}_3$	72x61x70x77	99.4%	7.1%	100%	5.0%	99.9%	3.7%	98.6%	11.0%
$\mathcal{N}_4$	65x99x87x23x31	98.4%	7.1%	100%	7.2%	99.8%	3.7%	98.4%	11.2%
$\mathcal{N}_5$	49x61x90x21x48	89.1%	11.4%	99.1%	9.6%	99.4%	4.9%	98.7%	9.1%
$\mathcal{N}_6$	97x83x32	100.0%	9.4%	100%	5.6%	100%	3.7%	100%	8.0%
$\mathcal{N}_7$	33x95x67x43x76	86.9%	8.8%	100%	7.2%	99.2%	3.8%	96%	12.0%
$\mathcal{N}_8$	78x62x73x47	99.8%	8.4%	100%	9.4%	100%	4.0%	100%	7.3%
$\mathcal{N}_9$	87x33x62	100.0%	12.0%	100%	10.5%	100%	5.0%	100%	6.7%
$\mathcal{N}_{10}$	76x55x74x98x75	86.7%	5.8%	100%	6.1%	98.3%	2.4%	93.9%	4.5%

2- DNN safety analysis

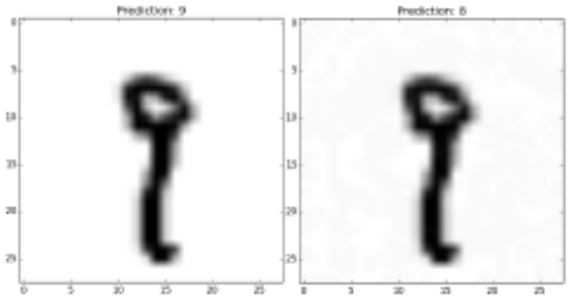
	$\sum_{1<k\leq K} s_k$	$\lvert\mathcal{O}(\mathcal{N})\rvert$	$\lvert\mathcal{T}\rvert$				
			cov_{SS}	(cov_{SS}^{w10})	cov_{DS}^{d}	cov_{SV}^{g}	$cov_{DV}^{d,g}$
$\mathcal{N}_1$	162	3490	3478	(949)	95	3490	95
$\mathcal{N}_2$	264	13780	13569	(2037)	205	13779	205
$\mathcal{N}_3$	290	14822	14739	(2170)	218	14820	215
$\mathcal{N}_4$	315	18072	17779	(2409)	250	18028	246
$\mathcal{N}_5$	279	11857	10567	(2120)	228	11780	227
$\mathcal{N}_6$	222	11027	11027	(1250)	125	11027	125
$\mathcal{N}_7$	324	16409	14264	(2593)	291	16275	280
$\mathcal{N}_8$	270	13263	13235	(1917)	192	13263	192
$\mathcal{N}_9$	192	5537	5537	(1050)	105	5537	105
$\mathcal{N}_{10}$	388	23602	20459	(2806)	312	23203	293

Table 4: Test suites generated for DNNs using different criteria

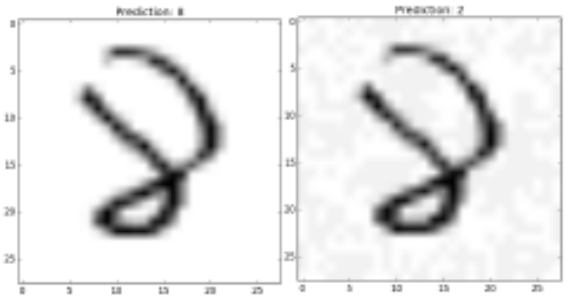
2-DNN safety analysis



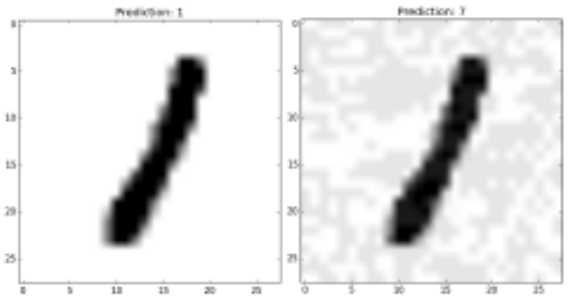
2- DNN safety analysis



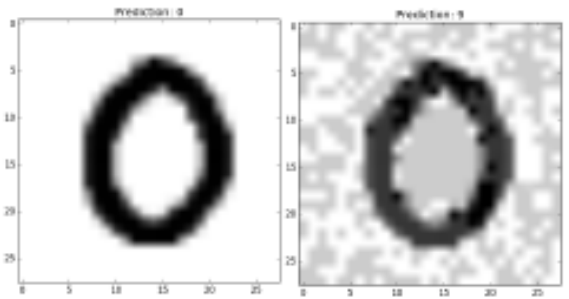
(a) 9 to 8 with $b = 0.02$



(b) 8 to 2 with $b = 0.05$

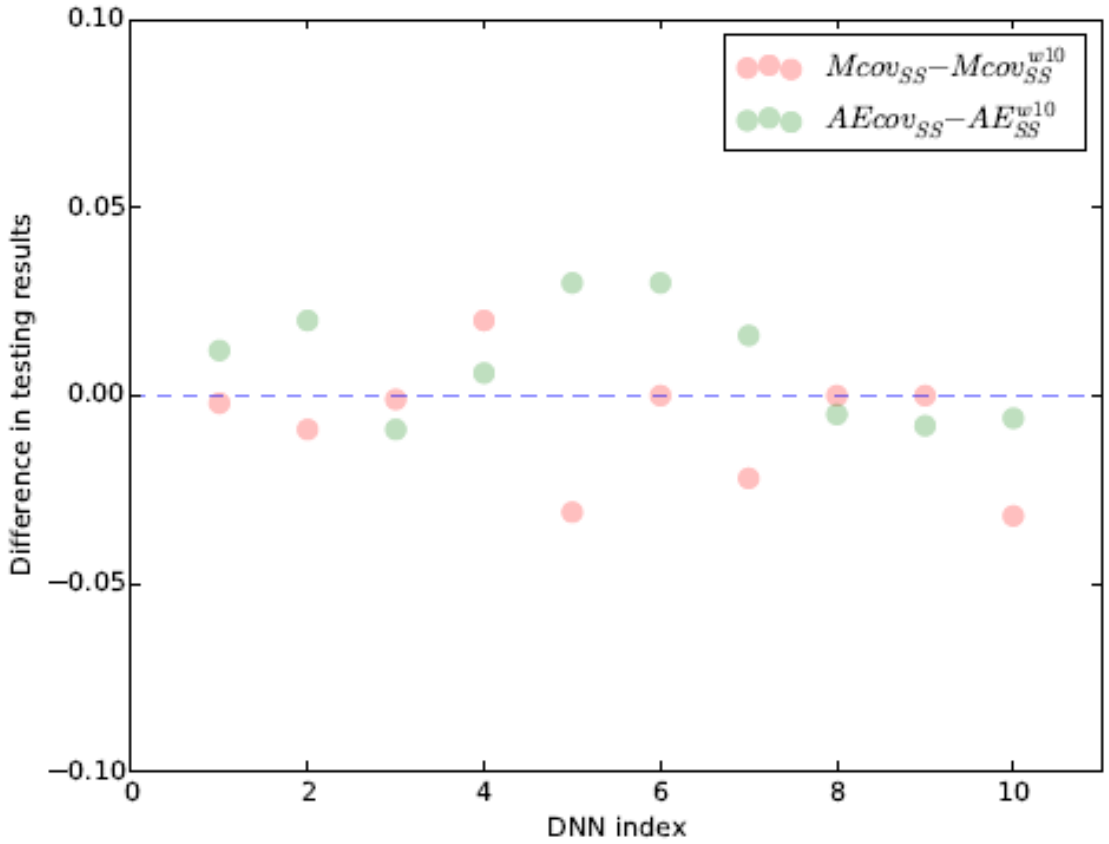


(c) 1 to 7 with $b = 0.1$

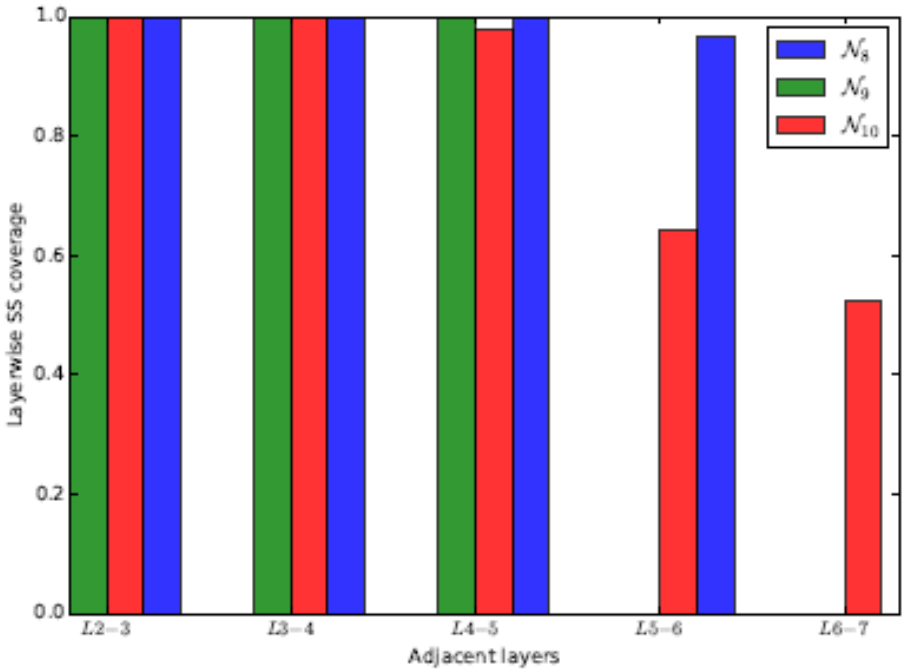


(d) 0 to 9 with $b = 0.2$

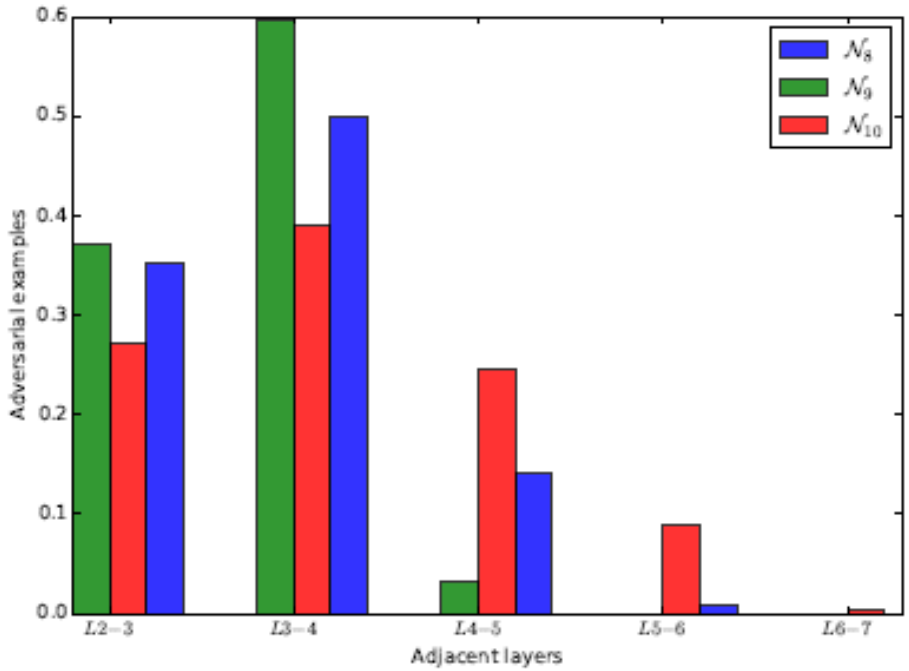
3- SS Cover with top weights



4- Layerwise behavior



(a) Layerwise coverage



(b) Adversarial examples at different layers

3- Cost of LP call

	$\mathcal{N}_8$			$\mathcal{N}_9$			$\mathcal{N}_{10}$		
	#vars	$\lvert\mathcal{C}\rvert$	t	#vars	$\lvert\mathcal{C}\rvert$	t	#vars	$\lvert\mathcal{C}\rvert$	t
$L2$-3	864	3294	0.58	873	3312	0.57	862	3290	0.49
$L3$-4	926	3418	0.84	906	3378	0.61	917	3400	0.71
$L4$-5	999	3564	0.87	968	3502	0.86	991	3548	0.75
$L5$-6	1046	3658	0.91	–	–	–	1089	3744	0.82
$L6$-7	–	–	–	–	–	–	1164	3894	0.94

4- Convolutional Neural Networks

	$Mcov_{SS}^{f_{1,1}}{}_{f_{2,1}}$	$AEcov_{SS}^{f_{1,1}}{}_{f_{2,1}}$	$Mcov_{SS}^{f_{1,1}}{}_{f_{2,2}}$	$AEcov_{SS}^{f_{1,1}}{}_{f_{2,2}}$
$\mathcal{N}_1^c$	99.7%	1.6%	99.7%	1.5%
$\mathcal{N}_2^c$	100%	7.6%	100%	6.8%