

Analysis of DNNs: On Test Coverage and Beyond

Divya Gopinath
(NASA AMES, KBR Inc.)

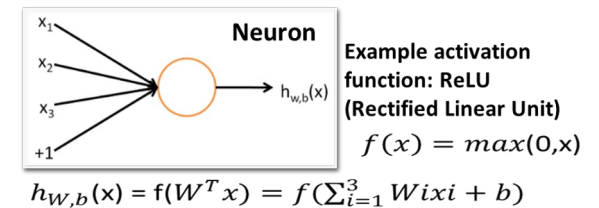
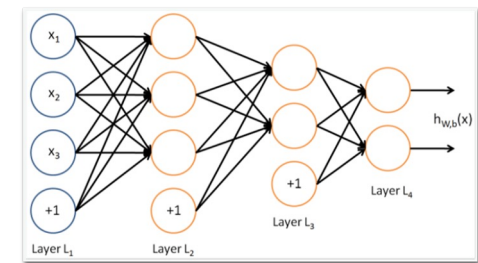
SafeDNN project (<https://ti.arc.nasa.gov/tech/rse/>)
RSE group, NASA AMES
Members: Corina S. Pasareanu, Divya Gopinath

Outline

- Why Testing Coverage is important for DNN-enabled systems?
- Overview of Existing Structural Coverage Metrics
- Evaluation of existing metrics using DNNCov tool
- Presenting a New Coverage Metric more meaningful for DNN testing
- Brief description of work beyond test coverage; feature-based view of DNN analysis enabling better software-engineering activities

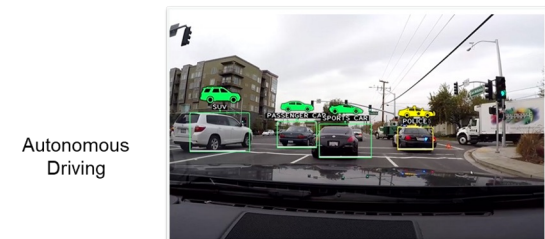
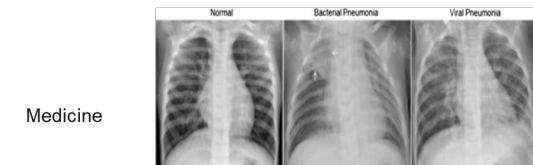
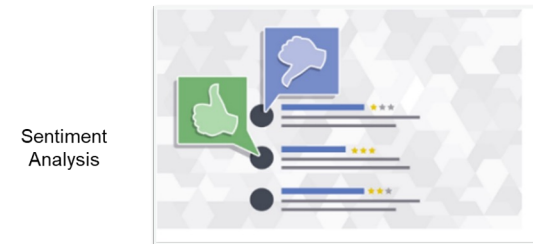
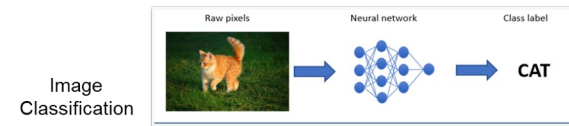
Deep Neural Networks

- Computing systems inspired by the biological NNs in animal brains
- Consist of neurons (computational units) organized in multiple layers
- Neurons can be active or not; last layer contains decisions
- Perform feature extraction and input transformation
- Learn (progressively improve performance) to do tasks by considering examples
- Can represent complex non-linear relationships



Applications

- Pattern analysis
- Image classification
- Sentiment analysis
- Speech/audio recognition
- Medical diagnosis
- Perception modules in self-driving cars



Safety and Security Concerns

- Hazardous cases in auto-pilot systems using DNNs
 - A Tesla car crashed into a trailer because the system failed to recognize the trailer as an obstacle due to its “white color against a brightly lit sky” and the “high ride height” [1]
 - A Google self-driving car recently crashed into a bus because it expected the bus to yield under a set of rare conditions but the bus did not [2]
- Lack of robustness
 - Small (imperceptible) changes to an input lead to misclassifications
 - Even for highly trained, highly accurate networks
- Lack of explainability
 - It is not well understood why a network gives a particular output
- Lack of specifications
 - Networks learn from examples, without high-level specifications
 - Networks are very large, highly interconnected structures; often have huge input spaces;
 - These issues make thorough verification/testing challenging
- What about the data?
 - Enough data? Poisoned/unreliable data? Bias?

[1] Who's responsible when an autonomous car crashes? <http://money.cnn.com/2016/07/07/technology/tesla-liability-risk/index.html>. (2016).

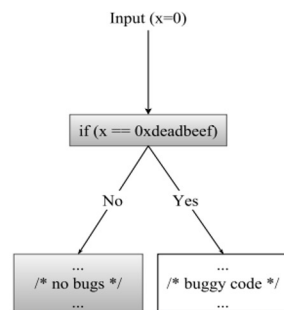
[2] Google's Self-Driving Car Caused Its First Crash. <https://www.wired.com/2016/02/googles-self-driving-car-may-caused-first-crash/> (2016)

The DNN Testing Landscape

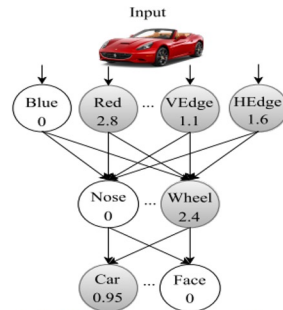
- Manual gathering and labelling of real-world data
 - Cumbersome and not sufficient
- Simulations to generate synthetic data
 - Often not representative of real world distributions
- Inspired from Traditional Software Testing:
 - Random testing (DeepHunter, Dlfuzz, TensorFuzz), Symbolic-execution based (HyDiff, Concolic Testing), Mutation testing (DeepMutation)
- DNN-specific:
 - DeepTest, DeepRoad: Generates erroneous driving scenes (blur/fog/rain) and different weather conditions
 - Black-box generation of images to cover different features (identified using techniques such as SIFT) (Feature-guided black-box testing [2017], Efficient and Effective Feature Space Exploration [2022])
 - Testing specific vulnerabilities such as adversarial inputs [C. E. Tuncali et. al., 2018], [I. J. Goodfellow et. al., 2014], [N. Papernot et. al., 2016]
- Main Challenge:
 - None of the existing approaches can cover more than a tiny fraction of the large input spaces of real-world

Why Testing Coverage?

- Performance of a DNN model is evaluated by calculating accuracy on a test suite
- Accuracy metric on a test set is not a sufficient measure of model quality
 - Portions of the network impacting behavior may not be adequately covered to show a difference in accuracy
- The effectiveness of testing and the reliability of the results depends on *test-suite adequacy*
 - Does it contain enough tests to cover different model behaviors, such that subtle vulnerabilities can be exposed?



(a) A program with a rare branch



(b) A DNN for detecting cars and faces

Test input with Red, Nose (clown) is a rare one that can be missed by test generation approaches

- Coverage Metrics
 - Typically used in software testing to measure adequacy of a test suite in covering different portions of code
 - Calculate the number of code entities (statements, branches, conditions, and so on) exercised by a test suite
 - A suite containing a larger % of these code-entities is considered more adequate than a suite with lesser coverage

Structural Coverage Metrics for DNN Testing

- Inspired by code coverage metrics, *structural coverage metrics* for DNN testing measure the portion of the structural entities (neuron activations, neuron output ranges, sets of neurons so on) exercised by a test suite
 - **DeepXplore**: Automated Whitebox Testing of Deep Learning Systems [SOSP 2017]: Proposes neuron coverage — counterpart for statement coverage
 - **DeepGauge**: Multi-Granularity Testing Criteria for Deep Learning Systems [ASE 2018]: More fine grained DNN coverage criteria
 - Effective White-Box Testing of Deep Neural Networks with Adaptive Neuron-Selection [ISSTA 2020]: Provides the **ADAPT** framework; focuses on generating new images for training.
 - **DeepHunter**: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks: Fuzzing framework for generating test images to increase different coverage metrics
 - Structural Test Coverage Criteria for Deep Neural Networks [TECS 2019]: Proposes **MC/DC** coverage for neural networks

Overview of some popular
Structural Coverage Metrics in DNN Testing

Neuron-level coverage

- **Neuron Coverage (NC):**

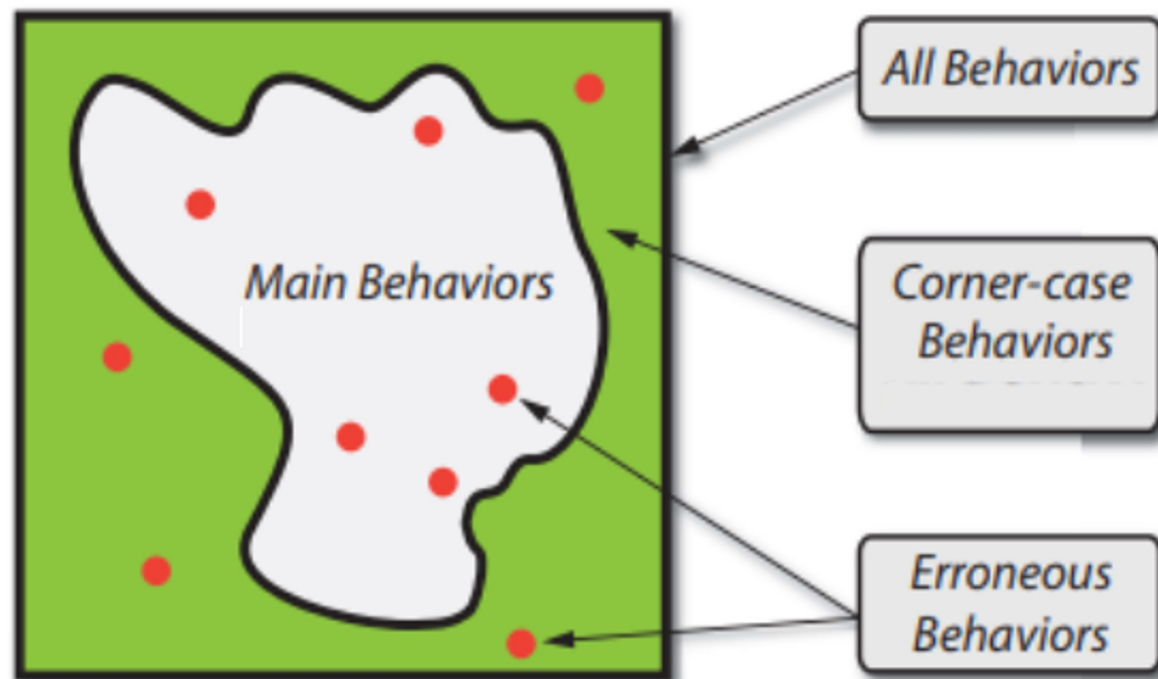
- Counterpart of statement coverage
- A neuron is considered covered, if its activation value is larger than 0 (or some specified threshold) for at least one test input
- What ratio of the total neurons have been activated beyond a specific threshold by the tests in the suite?

$$NC(T) = \frac{|\{n \in N \mid \exists \mathbf{x} \in T : \phi(\mathbf{x}, n) > 0\}|}{|N|}$$

Set of Neurons: $N = \{n_1, n_2, \dots\}$ **Set of Tests:** $T = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$

**Neuron output
for a given test:** $\phi(\mathbf{x}, n)$

Neuron Output Profile



Neuron output profile determined based on Training Data

Major Function coverage

- **K-Multisection Neuron Coverage (KMNC):**

- Measures how thoroughly the *major function regions* of neurons are covered by the tests in a suite
- Split each neuron's output into k equivalent sections,
 - Neuron output range : $[\text{low}_n, \text{high}_n]$
 - K equal sections: S_i^n :set of values in the i-th section, $1 \leq i \leq k$
- What ratio of these sections for a neuron are covered by some test

$$\text{KMNCov}(n, T, k) = \frac{|\{S_i^n \mid \exists \mathbf{x} \in T : \phi(\mathbf{x}, n) \in S_i^n\}|}{k}$$

- The overall KMNC metric for the model

$$\text{KMNCov}(T, k) = \frac{\sum_{n \in N} |\{S_i^n \mid \exists \mathbf{x} \in T : \phi(\mathbf{x}, n) \in S_i^n\}|}{k \times |N|}.$$

Corner Case coverage

- **Neuron Boundary Coverage (NBC):**

- Measures if tests in the suite covers the corner-case regions, indicating testing new (less-trained) behavior which could potentially be vulnerable
- What ratio of neurons have been activated above their highest and below their lowest training outputs?

$$\text{UpperCornerNeuron} = \{n \in N \mid \exists \mathbf{x} \in T : \phi(\mathbf{x}, n) \in (\text{high}_n, +\infty)\} \quad \text{LowerCornerNeuron} = \{n \in N \mid \exists \mathbf{x} \in T : \phi(\mathbf{x}, n) \in (-\infty, \text{low}_n)\}$$

$$\text{NBCov}(T) = \frac{|\text{UpperCornerNeuron}| + |\text{LowerCornerNeuron}|}{2 \times |N|}.$$

- **Hyperactive Neurons:**

- Recent research on DNN interpretability shows that neurons with large values potentially deliver more useful patterns impacting the network output

- **Strong Neuron Activation Coverage (SNAC):**

- Focuses more on hyperactive neurons
- What ratio of neurons have been activated above their highest training outputs?

$$\text{SNACov}(T) = \frac{|\text{UpperCornerNeuron}|}{|N|}$$

Layer-level coverage

- **Hyperactive Neurons at a layer:**

- For test $\mathbf{x}$, n_1 more hyperactive than n_2 if $\phi(\mathbf{x}, n_1) > \phi(\mathbf{x}, n_2)$
- Top K neurons at a layer are the ones having their output in the K-th largest values at that layer

- **Top K Neuron Coverage (TKNC):**

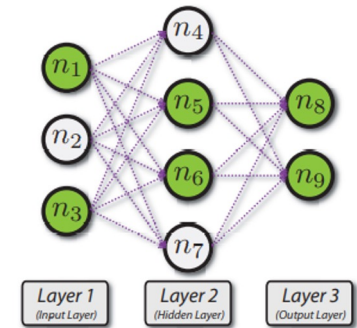
- Measures how many neurons have once been in the top-K list
- Ratio of Total # of top-K neurons at each layer by the Total # of neurons

$$\text{TKNCov}(T, k) = \frac{|\bigcup_{\mathbf{x} \in T} (\bigcup_{1 \leq i \leq l} \text{top}_k(\mathbf{x}, i))|}{|N|}$$

- **Top K Neuron Pattern Coverage (TKNP):**

- The patterns formed by sequences of the top-K neurons at each layer denote different kind of activated scenarios formed by the hyperactive neurons

$$\text{TKNPat}(T, k) = |\{(\text{top}_k(\mathbf{x}, 1), \dots, \text{top}_k(\mathbf{x}, l)) \mid \mathbf{x} \in T\}|.$$



Top K neurons:

Layer 1: {n1,n3}

Layer 2: {n5,n6}

Layer 3: {n8,n9}

Top K patterns:

{n1,n3},{n5,n6},{n8,n9}

n1-n5-n8

n1,n3-n5-n8

n1-n5,n6-n9

....

Modified Condition/Modified Decision (MC/DC) coverage

- Strict code coverage criteria used in safety critical software
 - Part of Avionics software development guidance DO-178B and C to ensure adequate testing of most critical (Level A) software
 - NASA requires 100% MC/DC coverage for any safety critical component
- MC/DC metric
 - If a choice can be made, all the possible factors (conditions) that contribute to that choice (decision) must be tested
 - Example: **Decision** ($(a > 3) \vee (b = 0)$) $\wedge (c \neq 4)$
 - Contains **conditions** $(a > 3)$, $(b = 0)$, $(c \neq 4)$
 - Each condition should evaluate to *true and false at least once*, and should *independently* affect the decision outcome
 - Test cases that achieve 100% MC/DC coverage
 - (1) $(a > 3) = \text{false}, (b = 0) = \text{true}, (c \neq 4) = \text{false}$
 - (2) $(a > 3) = \text{true}, (b = 0) = \text{false}, (c \neq 4) = \text{true}$
 - (3) $(a > 3) = \text{false}, (b = 0) = \text{false}, (c \neq 4) = \text{true}$
 - (4) $(a > 3) = \text{false}, (b = 0) = \text{true}, (c \neq 4) = \text{true}$

MC/DC for DNNs

- Each layer of a DNN transforms its input to represent it in terms of more abstract/complex features
- *Feature* learnt at a layer - single node or a subset of neurons
- Goal of MC/DC metric for DNN: test the impact of less complex features on more complex ones; impact of features (subsets of neurons) in one layer to the features in the next layer

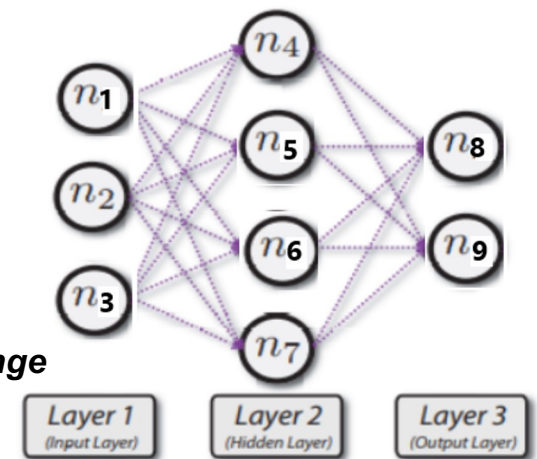
MC/DC metric definitions considering singleton features

SIGN CHANGE: $sc(n_{k,i})$ iff $sign(n_{k,i}, x_1) \neq sign(n_{k,i}, x_2)$ use a **sign change** in a neuron $n_{k,i}$ (i-th neuron at k-th layer)

$nsc(n_{k,i}, x_1, x_2)$: A pair of tests x_1 and x_2 cause a **no sign change** in a neuron $n_{k,i}$ (i-th neuron at k-th layer) iff $sign(n_{k,i}, x_1) = sign(n_{k,i}, x_2)$.

VALUE CHANGE: $vc(g, n_{k,i}, x_1, x_2)$: A pair of tests x_1 and x_2 cause a **value change** in a neuron $n_{k,i}$ wrt. some value function g , if $g(n_{k,i}, x_1, x_2) = true$

$\neg vc(g, n_{k,i}, x_1, x_2)$: A pair of tests x_1 and x_2 cause **no value change** in a neuron $n_{k,i}$ wrt. some value function g , if $g(n_{k,i}, x_1, x_2) = false$



MC/DC Coverage Metrics

- Coverage metrics are defined wrt. to feature pairs (features in adjacent layers)
- Let us consider singleton (single neuron) feature pairs

Sign-Sign Coverage: $sc(nk,i, x1, x2)$, $nsc(Nk \setminus nk,i, x1, x2)$, $sc(nk+1,j, x1, x2)$,
calculated for all neuron pairs of the DNN

Singleton Feature pairs: 20 { (n1,n4), (n1,n5), (n1,n6), (n1,n7), (n2,n4), (n2,n5), (n2,n6), (n2,n7), (n3,n4), (n3,n5), (n3,n6), (n3,n7), (n4,n8), (n4,n9), (n5,n8), (n5,n9), (n6,n8), (n6,n9), (n7,n8), (n7,n9) }

Tests: {x1,x2,x3}

Layer 1: {n1,n2,n3}: x1 = {1,2,-1}, x2 = {-1,2,0}, x3 = {-1,-2,0}

Layer 2: {n4,n5,n6,n7}: x1 = {3,2,-1,0}, x2={-2,4,0,1}, x3 = {-3,2,1,6}

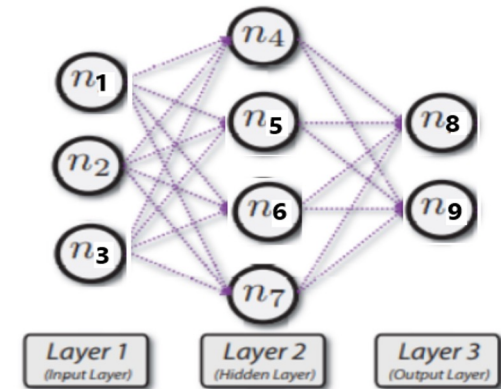
Layer 3: {n8,n9}: x1 = {1,2}, x2 = {-1,2}, x3 = {2,3}

Pairs satisfying SS (based on x1,x2): { (n1,n4) , (n1,n7) }

Pairs satisfying SS (based on x2,x3): { (n2,n6) , (n6,n8) }

Pairs satisfying SS (based on x1,x3): None

SS Coverage: (4/20) : 20%



MC/DC Coverage Metrics

Value-Sign Coverage: $vc(g, nk, i, x1, x2), nsc(Nk, i, x1, x2), sc(nk+1, j, x1, x2)$
calculated for all neuron pairs of the DNN

Sign-Value Coverage: $sc(nk, i, x1, x2), nsc(Nk \setminus nk, i, x1, x2), vc(g, nk+1, j, x1, x2), nsc(Nk+1, x1, x2)$
calculated for all neuron pairs of the DNN

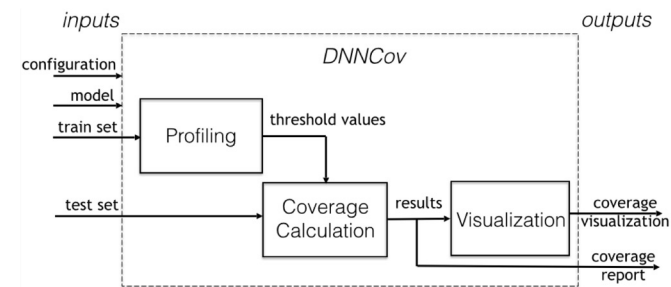
Value-Value Coverage: $vc(g, nk, i, x1, x2), nsc(Nk, x1, x2), vc(g, nk+1, j, x1, x2), nsc(Nk+1, x1, x2)$
calculated for all neuron pairs of the DNN

Evaluation of Structural Coverage Metrics

DNNCov

(An Overview of Structural Coverage Metrics for Testing Neural Networks [ISTTT 2022])

- **DNNCov** is an integrated framework that given a test-suite, calculates popular structural coverage metrics
 - NC (thresholds), KMNC, NBC, SNAC, TKNC, MC/DC
- Provides a quantitative coverage report and also a visualization of coverage
- This facilitates comparing different coverage metrics

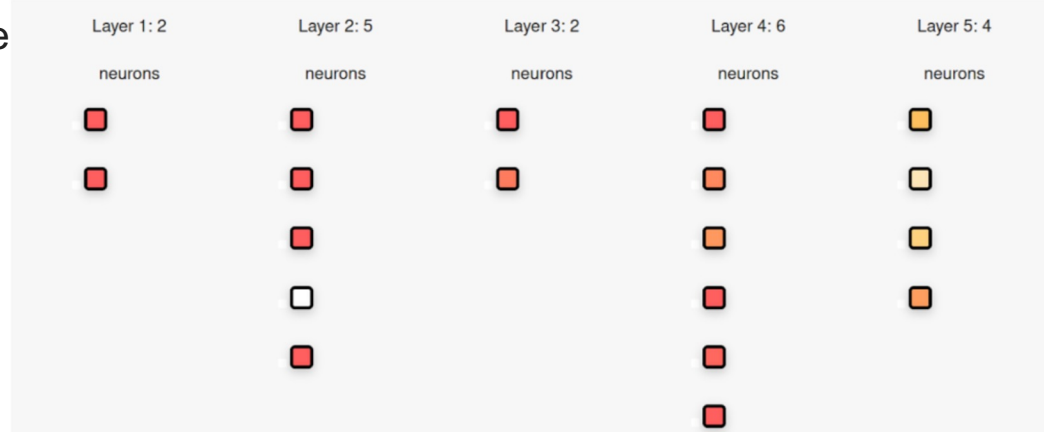


```
100%|██████████| 10000/10000 [02:11<00:00, 75.82it/s]

FINAL COVERAGES:
k-Multisection Neuron Coverage (k: 1000) = 59.1%
Top-k Neuron Coverage (k: 10) = 82.4%
Neuron Boundary Coverage = 0.58%
Strong Neuron Activation Coverage = 1.16%
Neuron Coverage (threshold: 0.75) = 67.05%
```

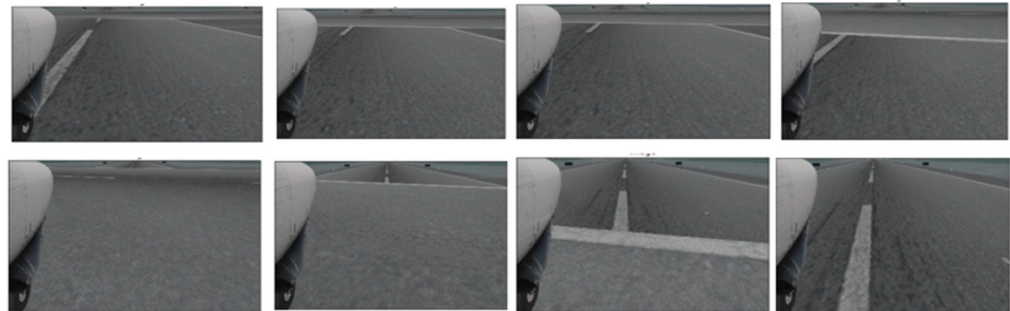
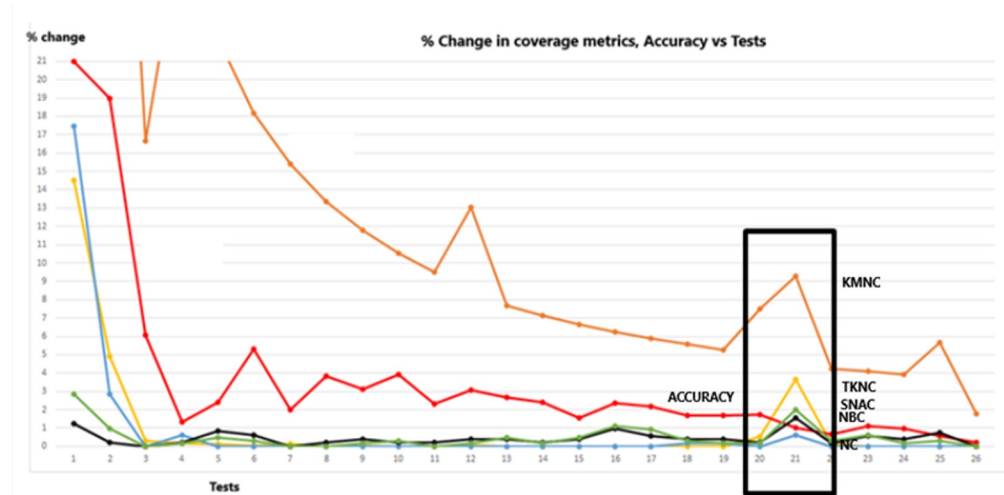
Quantitative Information

- The tool also displays quantitative information for every coverage obligation such as the minimum, maximum, total , variance in the number of tests
- It also has a web-based interface that helps *visualize coverage* by showing the architecture of the model and the coverage information per neuron
- This information can help understand which portions of the network have been exercised less and direct te



Experimental Evidence: Why Accuracy is not Enough?

- **TaxiNet** is a neural network takes a single picture of the runway as input and return the plane's position w.r.t. the center-line
- There may be tests which do not bring about much change to the accuracy metric but cover newer portions of the network, in turn, covering new scenarios
- On the other hand, we can also detect tests which are redundant in terms of their coverage



Evaluation Benchmarks

Model	Benchmark	#Dataset (training,test)	Test Accuracy	Model Architecture
LeNet-1	MNIST	(60k,10k)	90.60%	2 conv/2 maxpool
LeNet-4	MNIST	(60k,10k)	89.90%	2 conv/2 maxpool/1 dense
LeNet-5	MNIST	(60k,10k)	92.80%	2 conv/2 maxpool/2 dense
MNIST-Funct.	MNIST	(60k,10k)	96.34%	2 conv/4 act/1 maxpool/2 dense/1 flatten
MNIST-Pois.	MNIST	(60k,10k)	Test:98.63, Pois.:10.38%	2 conv/4 act/1 maxpool/2 dense/1 flatten
MNIST-Adv.	MNIST	(60k,10k)	Test: 97.87%, Adv.: 28.37%	2 conv/4 act/1 maxpool/2 dense/1 flatten
ResNet20	CIFAR-10	(50k,10k)	68.70%	21 conv/19 batchnorm/19 act/9 add /1 globalavgpool
TinyTaxiNet	TaxiNet	(51462,7386)	MAE (CTE:1.44, HE:2.75)	3 dense

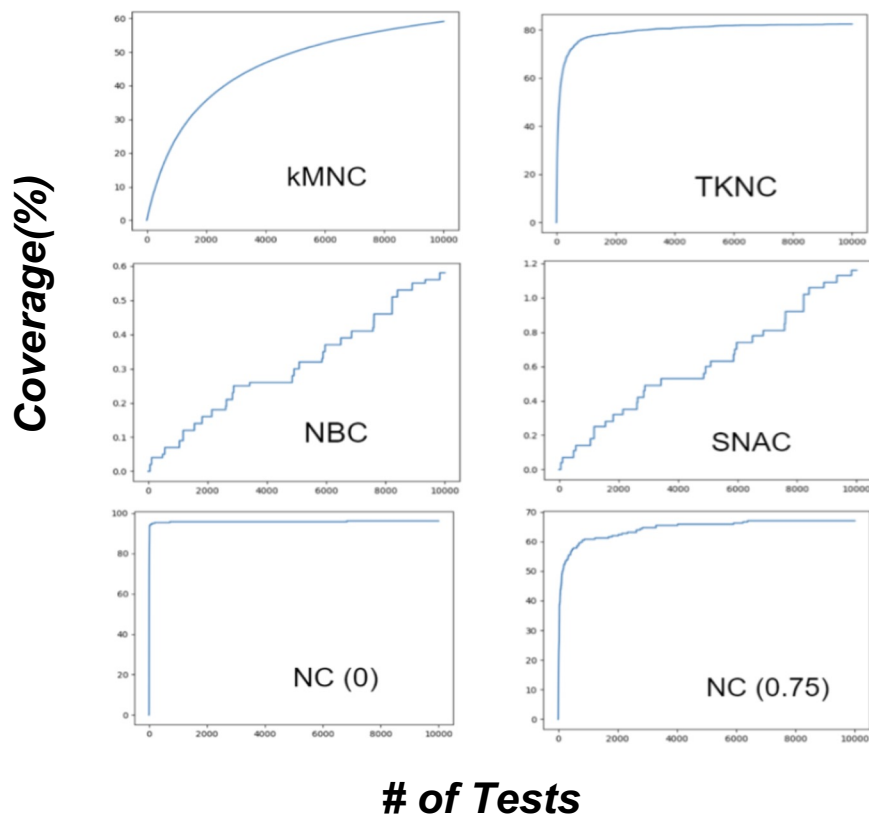
Comparison of Coverage Metrics

Coverage Metrics (%)	LeNet-1	LeNet-4	LeNet-5	ResNet20	TinyTaxiNet
KMNC (K: 10)	95.00	85.29	90.70	98.75	62.35
KMNC (K: 1000)	60.23	54.33	59.10	71.16	43.54
TKNC (K: 10)	88.57	81.59	82.40	65.09	52.06
TKNC (K: 1000)	1.00	3.27	4.93	3.90	0.59
NBC	0.87	0.66	0.58	5.55	2.01
SNAC	0.87	1.05	1.16	6.46	3.21
NC (Threshold: 0.00)	100.00	90.58	96.12	100.00	67.65
NC (Threshold: 0.20)	61.90	76.09	85.66	100.00	67.65
NC (Threshold: 0.50)	30.95	63.77	74.03	99.16	61.76
NC (Threshold: 0.75)	23.81	59.42	67.05	13.99	52.94
MC/DC (Sign-Sign)	5.77	44.17	10.62	24.66	27.60
MC/DC (Sign-Value)	63.46	35.68	8.40	51.58	28.65
MC/DC (Value-Sign)	23.08	58.20	13.71	52.20	41.67
MC/DC (Value-Value)	100.00	67.21	15.23	99.63	46.88

- Observations:

- NC(0) easiest to achieve, Low coverage of NBC and SNAC (same train and test distributions), MC/DC hard to achieve (more constraints)
- TinyTaxiNet model has low coverage than others (test set from simulations, whereas MNIST and CIFAR have standard test sets which cover the model adequately)

Coverage vs # of Tests (MNIST LeNet-5)



Coverage metric vs Functional Diversity

- A test-suite should test different functionalities of DNN model
 - DNN outputs as proxy for functional diversity
 - A test set covering 10 labels is better than a test that covers just 2 labels
- A good coverage metric should be sensitive to functional diversity of the test-suite
 - The coverage score for a test-suite covering more functionality should be higher than one that covers less
- Experiment: **MNIST test set**
 - Build test sets of same size with randomly selected inputs from 1 class (F1), 2 classes (F2), 3 classes (F3) so on
 - Determine coverage scores using all metrics for F1,F2,F3...F10
 - The metric that displays maximum change in scores from F1 through F10, could be considered the most sensitive to functional diversity

Coverage metric vs Functional Diversity

	Acc	KMNC (10)	KMNC (1000)	TKNC (10)	TKNC (1000)	NBC	SNAC	NC (0)	NC (0.2)	NC (0.5)	NC (0.75)	MC/DC (SS)	MC/DC (SV)	MC/DC (VS)	MC/DC (VV)
1 Label	97.00	63.55	28.68	34.85	6.46	0.05	0.08	100.00	98.65	92.30	72.57	0.00	0.00	75.23	99.70
2 Label	97.48	76.27	35.83	45.84	7.45	0.09	0.12	100.00	99.05	95.14	78.11	0.00	0.00	79.85	100.00
Difference	0.47	12.72	7.15	10.99	0.99	0.04	0.04	0.00	0.40	2.84	5.54	0.00	0.00	4.62	0.30
Normalized Diff	N/A	0.49	0.61	0.43	0.52	0.57	0.80	0.00	0.60	0.62	0.71	0.00	0.00	0.76	1.00
3 Label	95.93	79.74	37.11	51.12	7.80	0.08	0.08	100.00	99.05	95.54	79.33	0.00	0.00	81.04	100.00
Difference	-1.08	16.19	8.43	16.27	1.34	0.02	0.00	0.00	0.40	3.24	6.76	0.00	0.00	5.82	0.30
Normalized Diff	N/A	0.62	0.72	0.64	0.70	0.29	0.00	0.00	0.60	0.71	0.86	0.00	0.00	0.95	1.00
5 Label	96.30	85.13	39.16	57.74	8.15	0.11	0.10	100.00	99.32	96.08	79.46	0.00	0.00	81.19	100.00
Difference	-0.70	21.58	10.48	22.89	1.70	0.06	0.01	0.00	0.67	3.78	6.89	0.00	0.00	5.97	0.30
Normalized Diff	N/A	0.83	0.89	0.89	0.89	0.86	0.20	0.00	1.00	0.82	0.88	0.00	0.00	0.98	1.00
7 Label	96.65	87.32	39.89	58.54	8.25	0.10	0.12	100.00	99.05	96.49	80.27	0.00	0.00	81.34	100.00
Difference	-0.35	23.77	11.21	23.69	1.79	0.04	0.04	0.00	0.40	4.19	7.70	0.00	0.00	6.11	0.30
Normalized Diff	N/A	0.91	0.95	0.93	0.94	0.57	0.80	0.00	0.60	0.91	0.98	0.00	0.00	1.00	1.00
10 Labels	96.60	89.68	40.42	60.45	8.37	0.13	0.13	100.00	99.32	96.89	80.41	0.00	0.00	81.34	100.00
Difference	-0.40	26.12	11.74	25.60	1.91	0.07	0.05	0.00	0.67	4.59	7.84	0.00	0.00	6.11	0.30
Normalized Diff	N/A	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00

- Observations

Coverage metric vs Defect Detection Ability

- We expect higher coverage for test sets with inputs that reveal errors
- Adversarial scenario
 - We apply perturbation to MNIST images (FGSM)
 - Data sets A1, A2, .. A10 with increasing number of perturbed images
- Poison scenario
 - An attacker has “poisoned” the model by inserting an “input trigger”; when presented with an image with the trigger, the model misbehaves
 - Data sets P1, P2, .. AP10 with increasing number of poisoned images

Adversarial Scenario

	Acc	KMNC (10)	KMNC (1000)	TKNC (10)	TKNC (1000)	NBC	SNAC	NC (0)	NC (0.2)	NC (0.5)	NC (0.75)	MC/DC (SS)	MC/DC (SV)	MC/DC (VS)	MC/DC (VV)
Clean	98.22	90.26	45.29	44.30	6.25	0.10	0.12	98.65	97.30	90.41	45.27	0.00	1.49	21.27	50.75
Adversarial (1%)	97.26	92.06	45.50	44.43	6.30	1.81	3.21	98.65	97.30	90.41	45.27	0.15	1.72	22.28	50.75
Difference	-0.96	1.80	0.20	0.13	0.04	1.70	3.09	0.00	0.00	0.00	0.00	0.15	0.23	1.01	0.00
Normalized Diff	N/A	0.58	0.11	0.13	0.13	0.31	0.31	0.00	0.00	0.00	0.00	0.50	0.61	0.29	0.00
Adversarial (2%)	96.26	92.66	45.70	44.73	6.34	2.58	4.50	98.65	97.30	90.41	45.27	0.23	1.87	22.61	50.75
Difference	-1.96	2.41	0.41	0.43	0.09	2.48	4.38	0.00	0.00	0.00	0.00	0.23	0.38	1.34	0.00
Normalized Diff	N/A	0.78	0.23	0.43	0.28	0.45	0.44	0.00	0.00	0.00	0.00	0.77	1.00	0.38	0.00
Adversarial (3%)	95.32	92.85	45.89	44.81	6.38	3.07	5.55	98.65	97.30	90.41	45.27	0.23	1.87	23.21	50.75
Difference	-2.90	2.60	0.60	0.51	0.12	2.96	5.43	0.00	0.00	0.00	0.00	0.23	0.38	1.94	0.00
Normalized Diff	N/A	0.84	0.34	0.50	0.38	0.53	0.55	0.00	0.00	0.00	0.00	0.77	1.00	0.55	0.00
Adversarial (5%)	93.42	93.19	46.31	44.95	6.45	3.79	6.67	98.65	97.30	90.41	45.27	0.26	1.87	23.28	50.75
Difference	-4.80	2.93	1.02	0.65	0.20	3.69	6.55	0.00	0.00	0.00	0.00	0.26	0.38	2.01	0.00
Normalized Diff	N/A	0.95	0.58	0.64	0.63	0.66	0.66	0.00	0.00	0.00	0.00	0.87	1.00	0.57	0.00
Adversarial (7%)	91.40	93.34	46.67	45.14	6.51	4.67	8.12	98.65	97.30	90.41	45.27	0.30	1.87	24.48	50.75
Difference	-6.82	3.08	1.37	0.84	0.25	4.57	8.00	0.00	0.00	0.00	0.00	0.30	0.38	3.21	0.00
Normalized Diff	N/A	1.00	0.78	0.83	0.78	0.82	0.81	0.00	0.00	0.00	0.00	1.00	1.00	0.91	0.00
Adversarial (10%)	88.50	93.35	47.06	45.31	6.57	5.66	9.99	98.65	97.30	90.95	45.27	0.30	1.87	24.78	50.75
Difference	-9.72	3.09	1.76	1.01	0.32	5.56	9.87	0.00	0.00	0.54	0.00	0.30	0.38	3.51	0.00
Normalized Diff	N/A	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00	0.00

- Observations:

- NC(0, 0.2, 0.75), MC/DC (VV) report similar coverage across all datasets
- Other metrics show slight increase in coverage values with increasing number of errors

Poison Scenario

	Acc	KMNC (10)	KMNC (1000)	TKNC (10)	TKNC (1000)	NBC	SNAC	NC (0)	NC (0.2)	NC (0.5)	NC (0.75)	MC/DC (SS)	MC/DC (SV)	MC/DC (VS)	MC/DC (VV)
Clean	98.52	89.12	43.95	76.18	9.76	0.14	0.12	100.00	97.57	97.30	81.35	0.00	0.75	84.70	100.00
Poisoned (1%)	97.70	89.59	44.04	76.23	9.77	0.32	0.39	100.00	97.57	97.16	81.35	0.00	0.75	84.70	100.00
Difference	-0.82	0.47	0.09	0.05	0.01	0.18	0.27	0.00	0.00	-0.14	0.00	0.00	0.00	0.00	0.00
Normalized Diff	N/A	0.27	0.12	0.19	0.33	0.17	0.21	0.00	0.00	-1.00	0.00	0.00	0.00	0.00	0.00
Poisoned (2%)	96.74	89.87	44.13	76.10	9.77	0.49	0.57	100.00	97.57	97.30	81.35	0.00	0.75	84.55	100.00
Difference	-1.78	0.75	0.18	-0.08	0.01	0.35	0.44	0.00	0.00	0.00	0.00	0.00	0.00	-0.15	0.00
Normalized Diff	N/A	0.43	0.23	-0.30	0.33	0.34	0.34	0.00	0.00	0.00	0.00	0.00	0.00	-0.33	0.00
Poisoned (3%)	95.76	90.19	44.23	76.23	9.77	0.57	0.78	100.00	97.57	97.30	81.35	0.00	0.75	84.70	100.00
Difference	-2.76	1.07	0.28	0.05	0.02	0.44	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Normalized Diff	N/A	0.62	0.36	0.19	0.67	0.43	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Poisoned (5%)	93.96	90.47	44.36	76.12	9.78	0.77	0.86	100.00	97.57	97.30	81.35	0.00	0.75	84.85	100.00
Difference	-4.56	1.35	0.41	-0.06	0.03	0.63	0.74	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00
Normalized Diff	N/A	0.78	0.53	-0.22	1.00	0.61	0.57	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00
Poisoned (7%)	92.32	90.51	44.49	76.23	9.78	1.01	1.32	100.00	97.57	97.30	81.35	0.00	0.60	85.00	100.00
Difference	-6.20	1.39	0.54	0.05	0.03	0.87	1.20	0.00	0.00	0.00	0.00	0.00	-0.15	0.30	0.00
Normalized Diff	N/A	0.80	0.70	0.19	1.00	0.84	0.92	0.00	0.00	0.00	0.00	0.00	-1.00	0.67	0.00
Poisoned (10%)	89.60	90.85	44.72	75.96	9.79	1.17	1.43	100.00	97.57	97.30	81.35	0.00	0.75	84.70	100.00
Difference	-8.92	1.73	0.77	-0.22	0.03	1.03	1.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Normalized Diff	N/A	1.00	1.00	-0.81	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

- Observations:
 - Except KMNC, NBC, SNAC , all other metrics do not show sensitivity to the presence of poisoning attack

Discussion

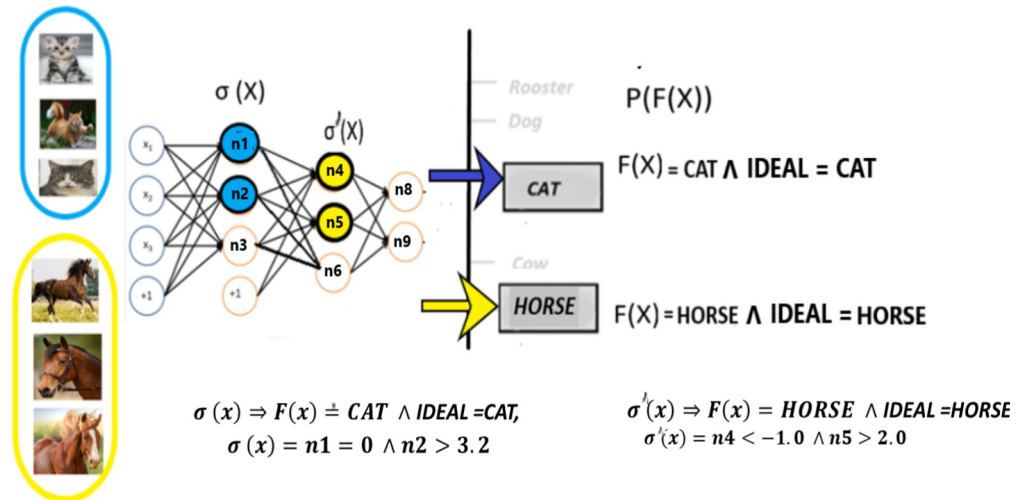
- Structural metrics are either too weak (NC) or too conservative and very time consuming (MC/DC)
- It is not clear if any of them show reliable correlation with functional diversity and defect detection
- A 60% KMNC score does not tell us how much functionality has been covered, how many input scenarios have been covered so on
- Coverage of structural entities do not have much meaning unless they can be clearly associated with different model behaviors, functionalities and input scenarios

Proposing a New Coverage Metric with more Semantic
Meaning

Neuron-Pattern based Rules as Meaningful Abstractions

(Rule-based Testing of Neural Networks (To appear D-SyMLe 2023))

- Given a DNN model and a set of inputs labelled with ground-truth labels (trainset) infer *rules* that abstract input-output behavior
 - $\sigma(X) \Rightarrow P(F(X))$, P is a property of the network function; $\sigma(X)$ is a pre-condition in terms of neuron constraints at a certain layer



- Rules for *correct classification to output label l*

$$\forall X, \sigma(X) \Rightarrow (F(X) = l \wedge l = l_{ideal})$$

$$\sigma(X) = \bigwedge_{N_i \in \mathcal{N}_L} N_i(X) \text{ op } V_i$$

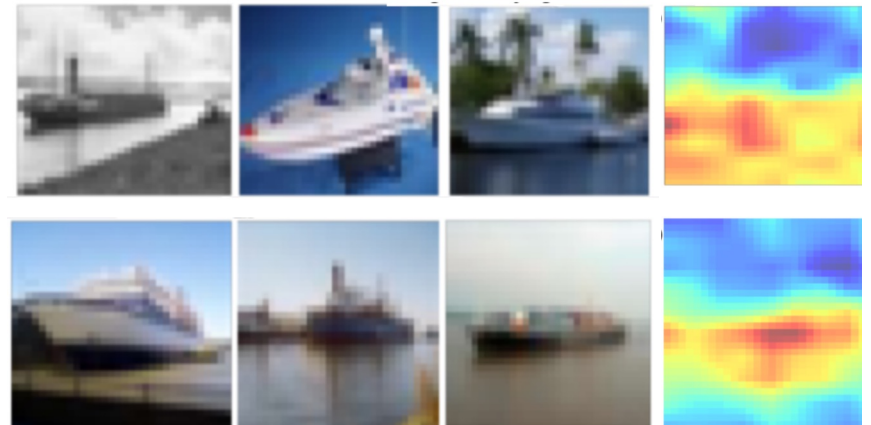
Neuron-Pattern based Rules for Different Input Scenarios

- Each $\sigma(X)$ (neuron pattern) *groups inputs that are similar to each other wrt. features* (ears, legs) that the model has extracted
- There can be more than one rules extracted for the same label
 - Representing different input scenarios for the same label
 - Model could use different combination of features for different scenarios

images satisfying rule for correct classification to deer



images satisfying two different rules for ship respectively



Neuron-Pattern based rules for mi-behavior

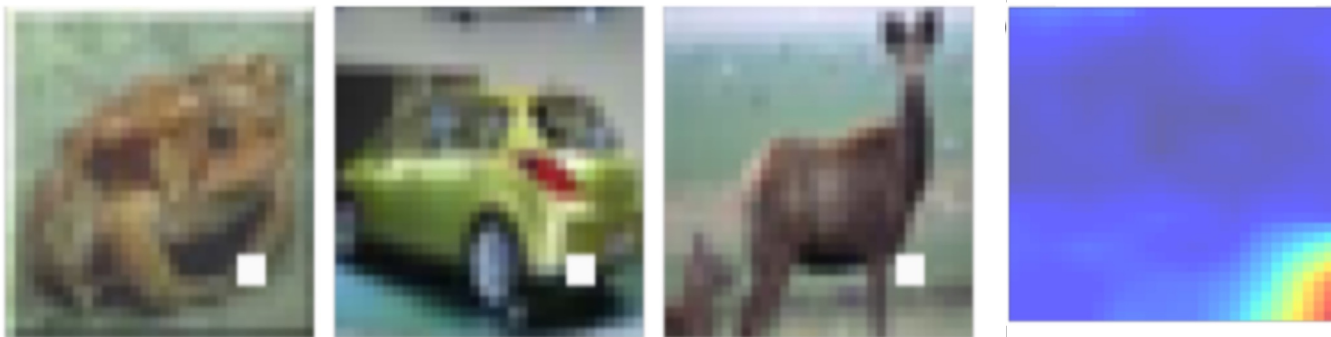
- Rules for mis-classification

$$\forall X, \sigma(X) \Rightarrow (F(X) \neq l_{ideal})$$

- Rules for specific vulnerabilities such as poison attacks

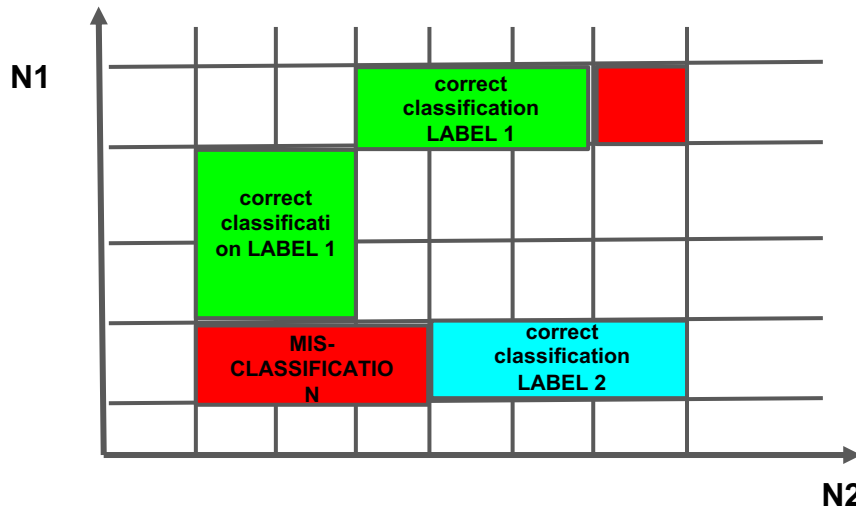
$$\forall X, \sigma_p(X) \Rightarrow (F(X) = p \wedge p \neq l_{ideal})$$

images satisfying mis-classification rule for poisoning (poison target horse)



Neuron-Pattern based Rules As Coverage Metrics

- The rules associate structural entities (groups of neurons, neuron patterns) to specific meaning with respect to network behavior
- The rules (by construction) covers different functionalities (output labels), input scenarios, incorrect behavior and specific vulnerabilities
- *Adequate coverage of the rules could translate to adequate testing of the model*



- Prophecy [ASE 2019] used to mine rules (uses dec-tree learning)
- To ensure good quality of rules, we filter out impure and overfitted rules
 - Select a subset of rules with precision > threshold on a separate validation set

New Coverage Metric vs Functional Diversity, Defect Detection Ability

- Extracted rules for correct classification to each of the 10 labels, mis-classification for an MNIST model

Test Suite	Metrics	Labels										
		Mis-classification	0	1	2	3	4	5	6	7	8	9
	Total # test obligations	62	56	70	138	123	133	141	91	160	145	180
Suite 1 (983 tests for labels 0, 9)	% test obligations covered	20.96	71.43	8.57	7.25	6.50	14.28	6.38	5.49	7.50	8.27	61.11
	Avg. tests per obligation	0.26	30.02	0.08	0.08	0.07	0.23	0.06	0.05	0.11	0.09	8.40
Suite 2 (983 tests across all labels)	% test obligations covered	22.58	55.36	38.57	44.20	37.39	42.85	39.71	59.34	46.87	50.34	36.66
	Avg. tests per obligation	0.22	5.78	5.60	2.57	2.25	2.67	2.03	3.57	2.37	1.90	1.90
Suite 3 (5000 Clean)	% test obligations covered	58.06	78.57	80.00	73.18	89.43	69.93	75.88	83.51	78.75	76.55	65.00
	Avg. tests per obligation	1.77	43.84	38.41	15.63	14.89	18.24	9.46	22.03	10.41	12.67	14.20
Suite 4 (2500 Clean, 2500 Adversarial)	% test obligations covered	72.58	32.14	61.43	80.43	80.49	68.42	70.21	86.81	77.50	82.06	64.44
	Avg. tests per obligation	4.66	2.72	28.00	13.81	14.62	13.68	11.80	13.16	8.23	8.25	11.56
Suite 5 (200 Passing)	% test obligations covered	1.61	14.28	24.28	21.01	20.32	18.79	21.27	18.68	16.87	20.00	12.22
	Avg. tests per obligation	0.03	11.25	4.71	2.93	3.72	2.08	2.70	3.47	3.29	1.75	15.90
Suite 6 (100 Passing, 100 Failing)	% test obligations covered	22.58	32.14	28.57	23.18	16.26	12.03	12.05	15.38	20.62	28.27	15.55
	Avg. tests per obligation	1.36	2.72	2.65	2.93	2.05	2.31	1.17	2.64	1.57	1.85	1.61

- Functional Diversity:
 - Determined the rules satisfied by the inputs in Test suite 1 (only labels 0,9) and Test suite 2 (all labels) with same size
 - The % rules (test obligations) per label is more evenly distributed for Suite 2 (SD: 7.02) vs Suite 1 (SD: 23/28)
- Defect Detection:
 - The % mis-classification rules satisfied by inputs in suite 4(adv+clean) > suite 3(clean) with same size
 - The % mis-classification rules satisfied by inputs in suite 6(pass+fail) > suite 5(pass) with same size

Supporting Other Testing Activities

- Test Oracles:
 - Labeling new test data is a laborious process
 - Rules for correct classification to a label can be used as ground-truth to label new test inputs
 - Rules for correct vs incorrect behavior can be used to label a subset of tests as passing/failing without execution, saving testing time
- Test generation, Test-suite reduction/selection/prioritization

Future Work

- Coverage and Precision of rules tied to the distribution of the training dataset and model accuracy
 - Formally characterize and quantify portions not covered, yet relevant
 - Combining with other structural metrics (ex. NBC) or Surprise Adequacy [Kim et. al. 2018]
 - Refine rules using sample OOD test data
- Use of coverage results for certification
 - In software development, system-level requirements are decomposed to input-output specifications of the software modules, translated to source code
 - Direct traceability of code coverage to system-level requirements
 - Not straight-forward in DNN systems, due to the data-driven nature of implementation
 - Associating coverage of structural entities to model behavior, input features and vulnerabilities helps to some extent
- Feature-guided Analysis of DNNs (To Appear, FASE 2023)
 - Mine rules corresponding to domain-spec features (time of day, weather, objects on the scene) that appear in system-level requirements
 - Coverage of such rules could enable easier traceability
 - Runtime checking of conformance to ODDs, OOD data
 - Formal Verification of system-level requirements

Thank You!

Brief Overview of other Coverage Metrics

- Surprise Adequacy, Safety coverage