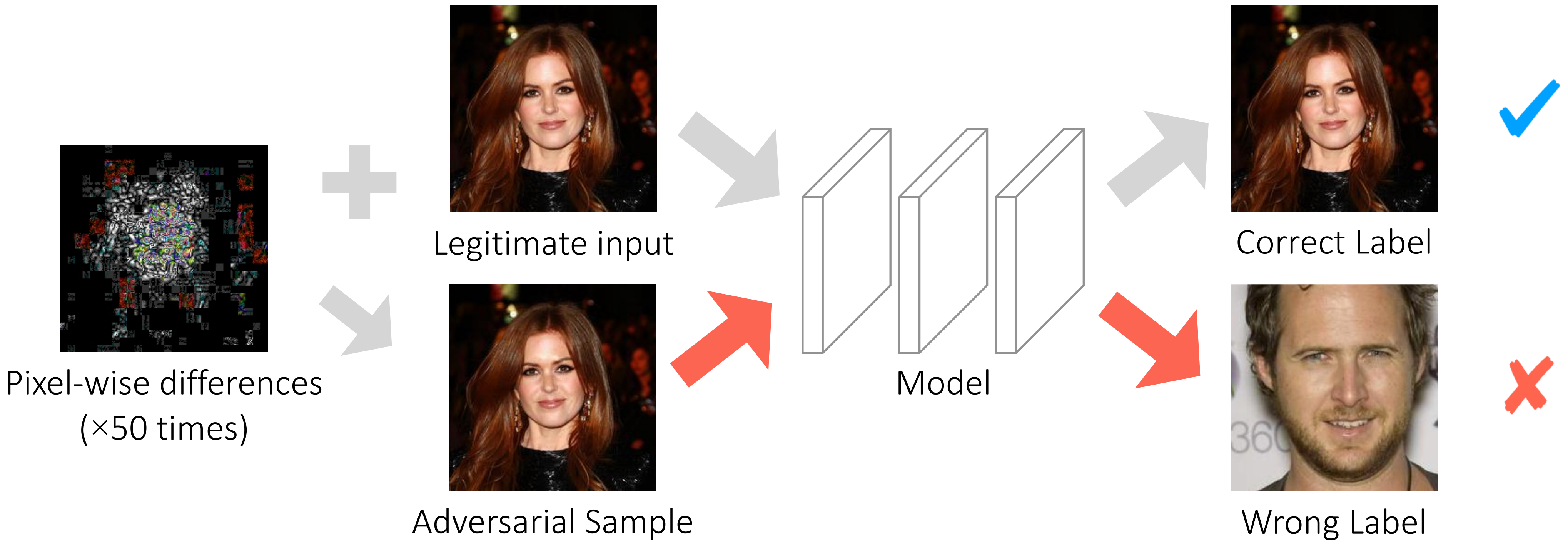
NIC: Detecting Adversarial Samples with Neural Network Invariant Checking

Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, Xiangyu Zhang

PURDUE UNIVERSITY

Adversarial Samples in Deep Neural Networks

- Adversarial samples are model inputs generated by adversaries to fool neural networks

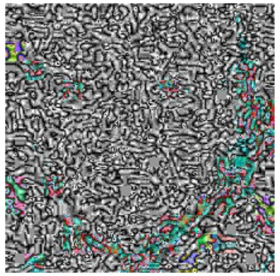


Existing Adversarial Attacks

- **Semantic based perturbations**
  - Change a/a few region(s) of the image
  - Simulate real world scenarios

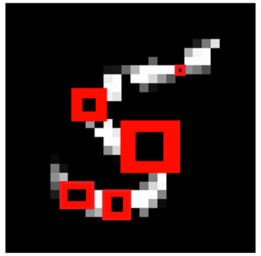
- **Pervasive perturbations**
  - Alter images in pixel level
  - Different distance metrics: L_0, L_2, L_∞

$$\Delta(x, x') = ||x - x'||_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{\frac{1}{p}}$$



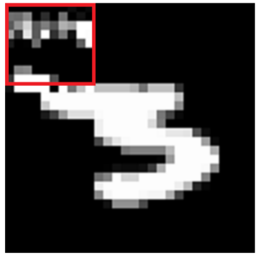
Representative Attacks

- Semantics based perturbations



Dirt: Camera lens have dirt.

Brightness: Different lighting conditions.



Rectangle: Camera lens are blocked by another object.

Trojan: Watermarks.

- Pervasive perturbations

FGSM

BIM

$C\&W_i$

$C\&W_2$

DeepFool

JSMA

$C\&W_0$

Existing Detection Methods

- Many detection and defenses have been proposed, we just list a few detection approaches here

- Characterizing the dimensional properties of adversarial regions
  - LID from ICLR 2018, oral presentation

- Denoisers that can remove/reform perturbations
  - MagNet from CCS 2017
  - HGD from CVPR 2018
    - First place in the NeurIPS 2017 competition on defense against adversarial attacks

- Prediction inconsistency
  - Feature Squeezing from NDSS 2018

However … …

- We found that most existing detection methods work on a subset of existing attacks or datasets (will show in evaluation later)
- Similar results are found by other researchers

All (evaluated) detection methods show comparable discriminative ability against existing attacks. Different detection methods have their own strengths and limitations facing various kinds of adversarial examples.[0]

- This is not new in security. E.g., defenses/detection to attacks that redirect the flow of execution of a program

[0]: Xiang Ling, Shouling Ji, Jiaxu Zou, Jiannan Wang, Chunming Wu, Bo Li and Ting Wang, DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Models, IEEE S&P, 2019

Basic idea: Invariants

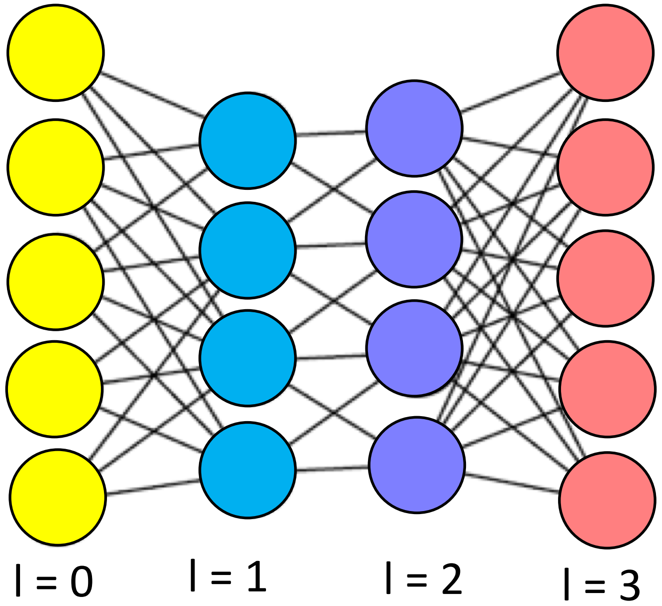
```
01: def fib(n):
02:   assert(n>=0)
03:   assert(from line 6 or 10)
04:   if n == 0 or n == 1:
05:     return n
06:   return fib(n-1)+fib(n-2)
07:
08: def main():
09:   x = input('Input a number:')
10:   print fib(x)
```

Key idea: using invariant checks to allow correct behaviors and forbidden other possible (malicious) behaviors.

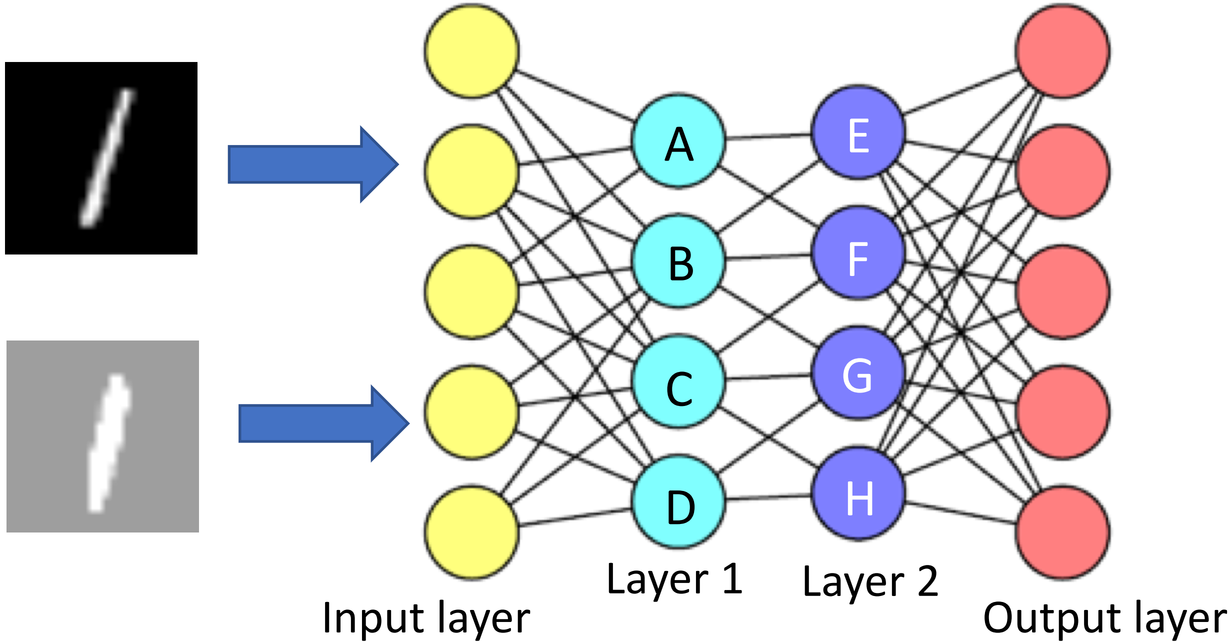
DNN Invariants

- Value invariants
  - Possible neuron value distributions of each layer
- Provenance invariants
  - Possible neuron value patterns of two consecutive layers
- If one input violates one invariant, it is detected as an adversarial sample

```
def DNN():
  for l in model.layers():
    if(l==0): x_l = input
    else: x_l+1 = f_l(w_l*x_l+b_l)
```

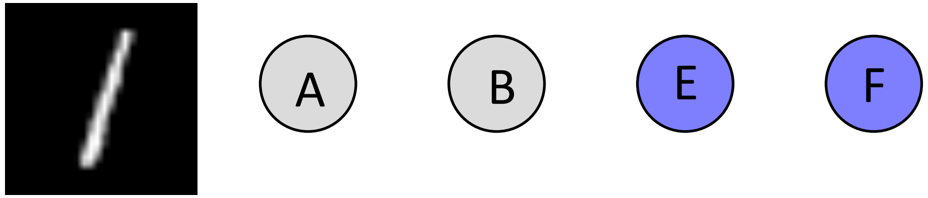


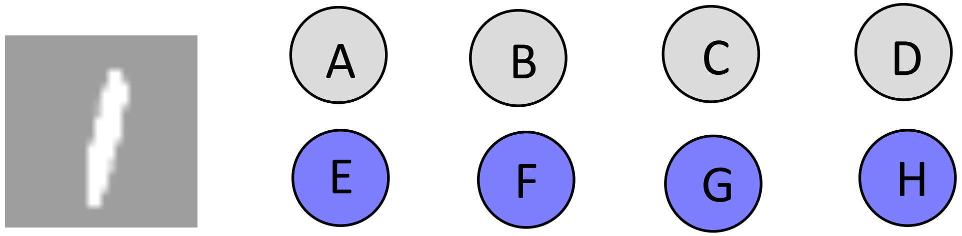
DNN Value Invariant



Activation of a benign sample of digit 1:
<A: 20, B: 30, C: 0, D, 0> <E: 40, F: 20, G: 0, H: 0>

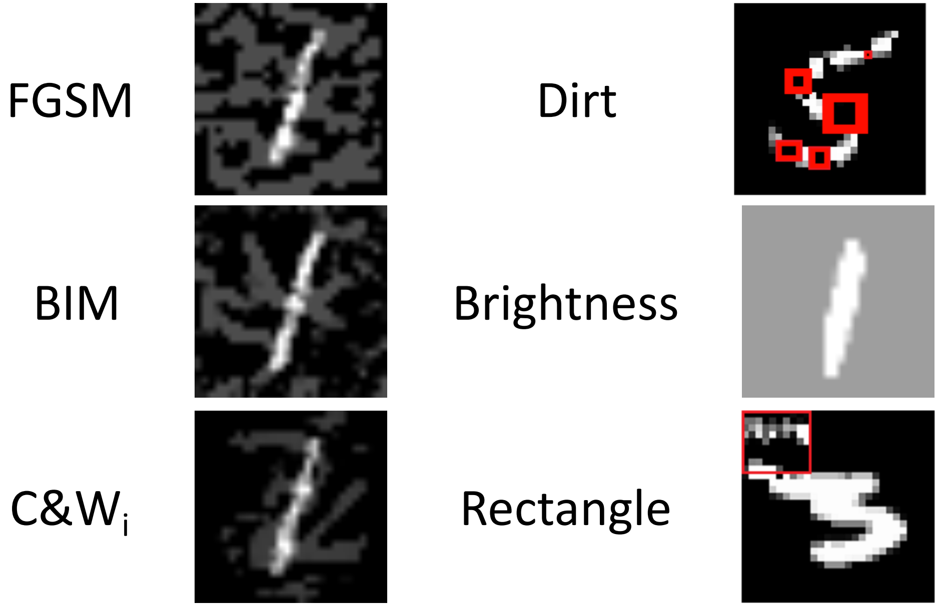
Activation of an adversarial sample of digit 1:
<A: 10, B: 10, C: 10, D, 10> <E: 10, F: 11, G: 9, H: 10>



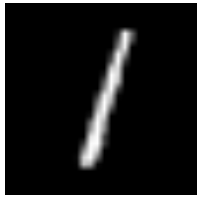


DNN Value Invariant

- Perturbations will change the activation patterns
  - Many attacks violate value invariants
  - Most attacks have new activation patterns in hidden layers
- Value Invariant
  - Trained classifiers that capture the activation patterns (of benign input samples) in each layer



Train DNN Value Invariant



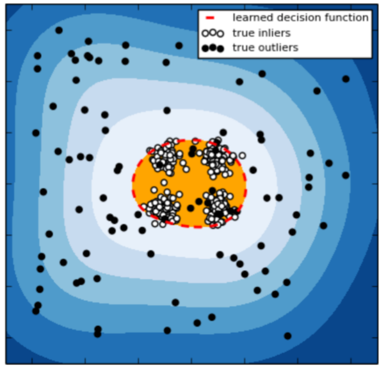
Activation of a benign sample of digit 1:
<A: 20, B: 30, C: 5, D, 5> <E: 40, F: 20, G: 3, H: 3>

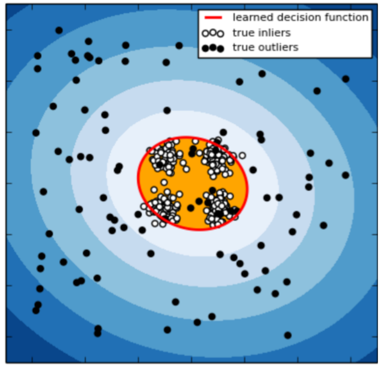
Activation of an adversarial sample of digit 1:
<A: 10, B: 10, C: 10, D, 10> <E: 10, F: 11, G: 9, H: 10>

1. Dump all neuron values in one layer using all benign samples

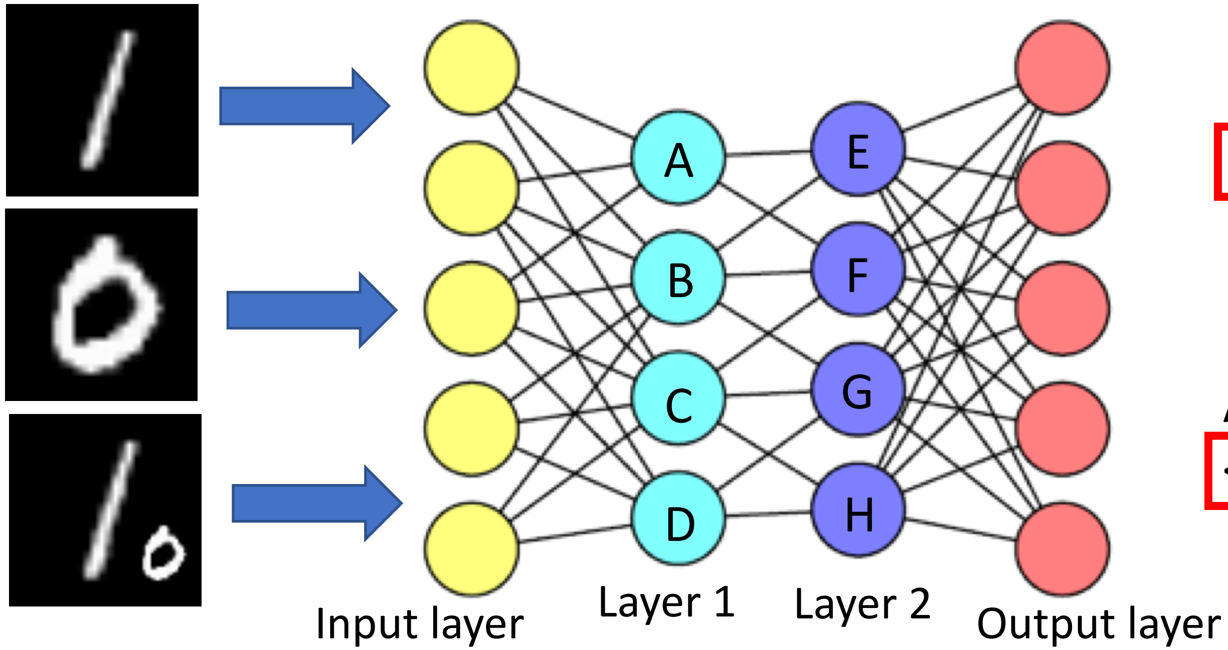
Sample 1:<A: 20, B: 30, C: 05, D, 05>
Sample 2:<A: 40, B: 20, C: 15, D, 13>
Sample 3:<A: 30, B: 34, C: 35, D, 52>

2. Train a classifier for each layer (One-class SVM)





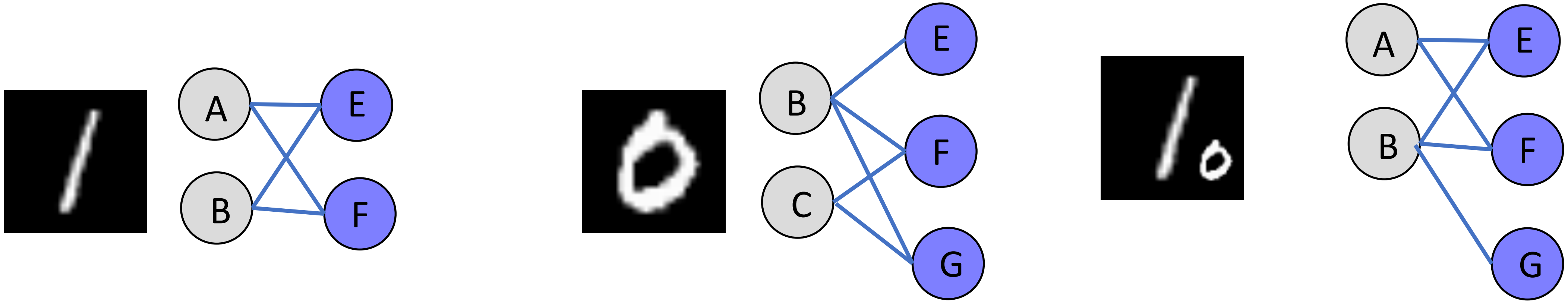
DNN Provenance Invariant



Activation of a benign sample of digit 1:
<A: 20, B: 30, C: 0, D: 0><E: 40, F: 20, G: 0, H: 0>

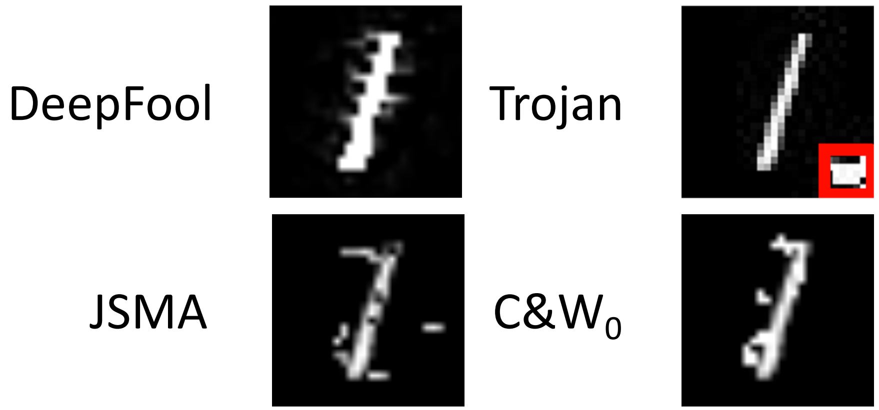
Activation of a benign sample of digit 0:
<A: 0, B: 30, C: 40, D: 0><E: 30, F: 60, G: 50, H: 0>

Activation of an adversarial sample:
<A: 22, B: 34, C: 0, D: 0><E: 28, F: 54, G: 46, H: 0>

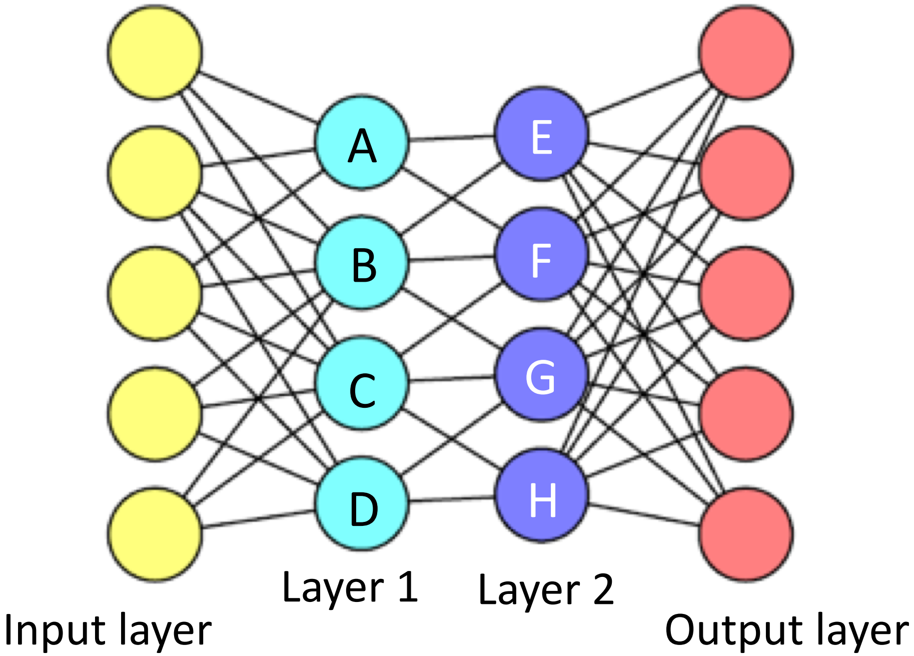


DNN Provenance Invariant

- DNN model may focus on different parts of the input in different layers
  - The patched image looks similar to 1 in layer 1 and look similar to 0 in layer 2
  - Using individual value invariants can not detect such attacks
- Provenance invariant
  - Trained classifiers that capture the activation patterns across two consecutive layers



Train Provenance Invariant



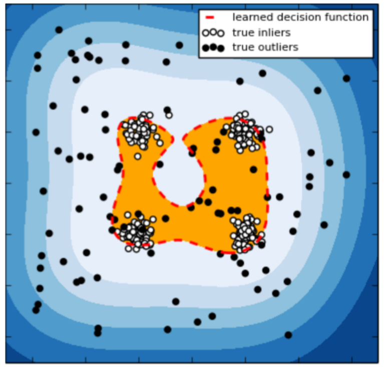
- Train on raw neuron values
  - Too many of them, hard to train
- Lower the dimensions of individual layers

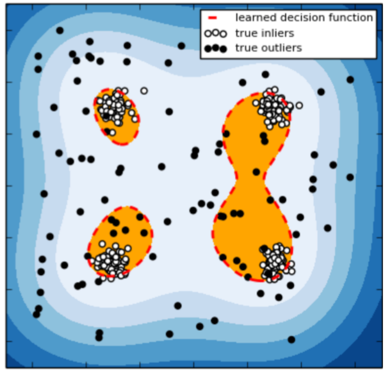
1. Lower the daemons of neurons in hidden layers

Sample 1:<A: 0.3, B: 0.5, C: 0.1, D: 0.1>
Sample 1:<E: 0.6, F: 0.2, G: 0.1, H, 0.1>
Sample 2:<A: 0.2, B: 0.4, C: 0.3, D: 0.1>
Sample 2:<E: 0.3, F: 0.4, G: 0.2, H, 0.1>

2. Train a classifier on 2 consecutive layers

Sample 1:<A: 0.3, B: 0.5, C: 0.1, D: 0.1,
E: 0.6, F: 0.2, G: 0.1, H, 0.1>
Sample 2:<A: 0.2, B: 0.4, C: 0.3, D: 0.1
E: 0.3, F: 0.4, G: 0.2, H, 0.1>





Evaluation

- Hardware
  - One server with two Xeon E5-2667 2.3GHz 8-core processors, 128 GB of RAM, 2 Tesla K40c GPU, 2 GeForce GTX TITAN X GPU and 4 TITAN Xp GPU cards.
  - The other one has two Intel Xeon E5602 2.13GHz 4-core processors, 24 GB of RAM, and 2 NVIDIA Tesla C2075 GPU cards
- Comparison with others
  - LID
  - MagNet
  - HGD
  - Feature Squeezing

Evaluation

- Datasets and models
  - MNIST: Cleverhans (*2), Carlini's model from IEEE S&P 2017, LeNet-4/5
  - CIFAR-10: Carlini's model, DenseNet
  - ImageNet: ResNet50, VGG19, Inceptionv3, MobileNets
  - LFW: VGG19 (Trojan attack)
- Attacks
  - FGSM, BIM, C&W attacks, DeepFool, JSMA
  - Dirt, Brightness, Rectangle, Trajon
  - Parameters adopted from previous papers (e.g., Feature Squeezing)

Results

- NIC achieves over 90% detection accuracy on all attacks
- Other methods achieve good results on a subset but fail to work on some of them (low detection accuracy)
  - LID: Good at L_∞ attacks on MNIST and CIFAR, but poor performance large sized images (e.g., ImageNet)

Method	Dataset	FGSM (L_∞)	DeepFool (L_2)	C&W (L_0)
NIC	CIFAR	100%	100%	100%
	ImageNet	100%	90%	100%
LID	CIFAR	94%	84%	90%
	ImageNet	82%	83%	79%

Results

- NIC achieves over 90% detection accuracy on all attacks
- Other methods achieve good results on a subset but fail to work on some of them (low detection accuracy)
  - MagNet: it does not perform well on many $L_{0/2}$ attacks, and it is hard to train on large sized image datasets

Method	Dataset	FGSM (L_∞)	C&W (L_2)	JSMA (L_0)
NIC	MNIST	100%	100%	100%
	CIFAR	100%	100%	100%
MagNet	MNIST	100%	87%	84%
	CIFAR	100%	89%	74%

Results

- NIC achieves over 90% detection accuracy on all attacks
- Other methods achieve good results on a subset but fail to work on some of them (low detection accuracy)
  - HGD: denoisers designed to work on large sized image datasets, but does not work well on L_0 and L_2 attacks

Method	Dataset	FGSM (L_∞)	DeepFool (L_2)	C&W (L_0)
NIC	ImageNet	100%	90%	100%
HGD		97%	83%	82%

Results

- NIC achieves over 90% detection accuracy on all attacks
- Other methods achieve good results on a subset but fail to work on some of them (low detection accuracy)
  - Feature Squeezing: not good at L_∞ attacks on large sized images (e.g., ImageNet) and some patching attacks

Method	Dataset	FGSM (L_∞)	C&W (L_2)	C&W (L_0)
NIC	MNIST	100%	100%	100%
	ImageNet	100%	90%	100%
Feature Squeezing	MNIST	100%	100%	94%
	ImageNet	43%	92%	98%

Results

- NIC achieves over 90% detection accuracy on all attacks
- Other methods achieve good results on a subset but fail to work on some of them (low detection accuracy)
  - Feature Squeezing: not good at L_∞ attacks on large sized images (e.g., ImageNet) and some patching attacks

Method	Dataset	Trojan	Brightness	Dirt	Rectangle
NIC	MNIST	100%	100%	100%	100%
	LFW	100%	100%	100%	100%
Feature Squeezing	MNIST	82%	39%	97%	72%
	LFW	67%	40%	89%	82%

Other results (summary)

- One-class Classifiers
  - One-class SVM works better in tested methods
- Value invariant or provenance invariant
  - They are complementary to each other!
  - Using just one of the two cannot get good results on all attacks
- Adaptive attacks
  - Adversary knows the original model, the detection methods, and all the value invariants and provenance invariants
  - $C\&W_2$ based attack
  - 97% success rate on MNIST and CIFAR-10
  - For MNIST, L_2 distortion is 3.98 (Feature Squeezing is 2.80)

Conclusion and Future Work

- We proposed a DNN invariant based adversarial sample detection method, NIC inspired by program invariants
- NIC only needs benign training samples, is able to detect a very large set of existing attacks, and achieves better performance than all the compared existing work
- NIC increases the bar for adaptive adversary, but is still vulnerable to such adaptive attacks
- Future work
  - Improve invariants quality
  - Adaptive adversary
  - Harden the model instead of just detecting adversarial samples
  - Etc…

Thank you!

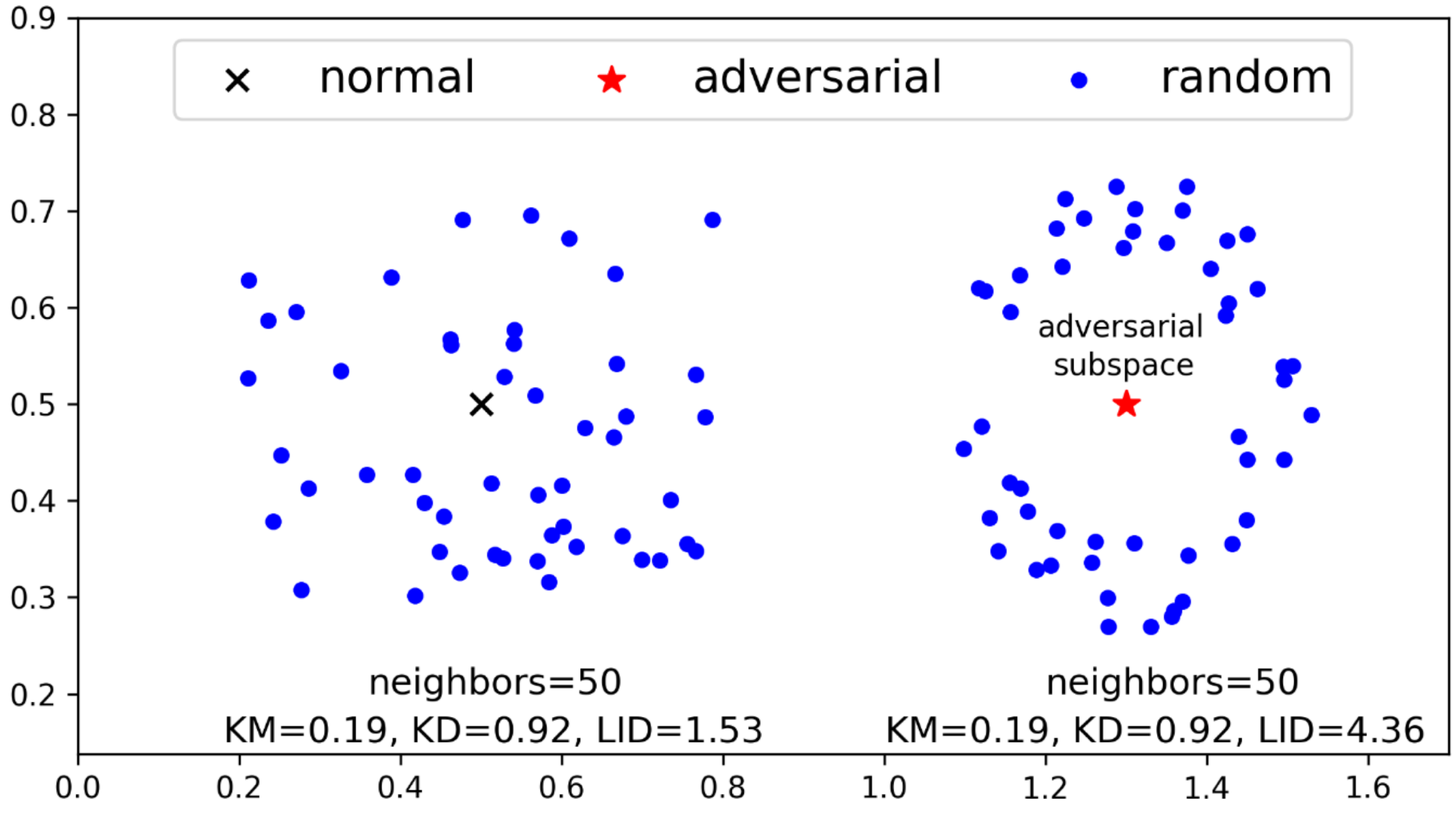
Existing Adversarial Attacks

- Targeted attacks
  - Manipulate models to output a specific classification label
  - More catastrophic
  - $\min \Delta(x', x)$ **s.t.** $x, x' \in X \wedge \boxed{C(x') \neq C(x)} \wedge \boxed{C(x') = t}$
- Untargeted attacks
  - Lead to misclassification without a target
  - $\min \Delta(x', x)$ **s.t.** $x, x' \in X \wedge \boxed{C(x') \neq C(x)}$

X : input space
$C(\cdot)$: classifier
t : target label

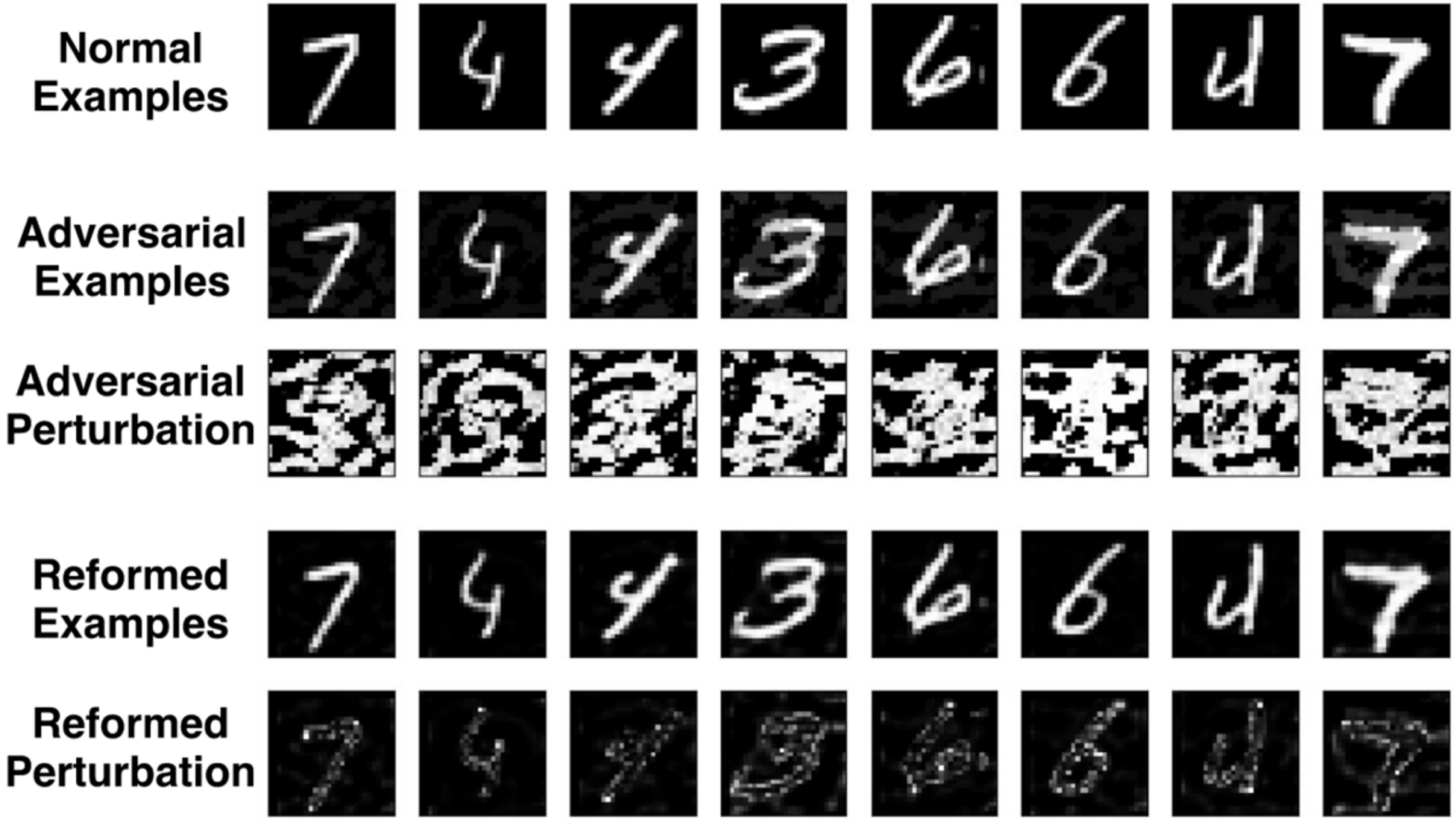
Existing detection

- Characterizing the dimensional properties of adversarial region (LID from ICLR'18)



Existing detection

- Using denoisers to remove perturbations (MagNet from CCS 17)



Existing detection

- Prediction inconsistency (Feature Squeezing from NDSS 2018)