Robustness of Deep Autoencoders in Intrusion Detection under Adversarial Contamination

Pooria Madani and Natalija Valjic

EECS at York University

HoTSoS, April ’18

Machine Learning as a strategy

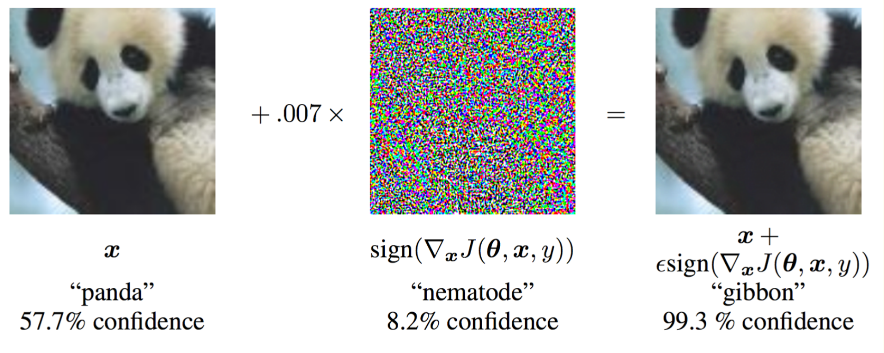
- “security is sometimes thought of as a chess game between two players. For a player to win, it is not only necessary to have an effective strategy, one must also anticipate the opponent’s response to that strategy.” [Huang *et. al. 2011*]
  - Should we anticipate that adversaries would try to cause our machine learning algorithms to fail in many ways?

- In many cybersecurity applications (including intrusion detection) modelled phenomenon are not **stationary**.
  - Normal-user/Adversary behavior changes over time.
  - We are required to frequently retrain our learning-based detection models to cope with moving concepts.

- Thus, the lack of stationarity and frequent retraining provide great opportunity for sophisticated adversaries to **poison** the learning process – sometimes in a targeted manner.

ML cannot be treated as a black-box in cybersecurity!

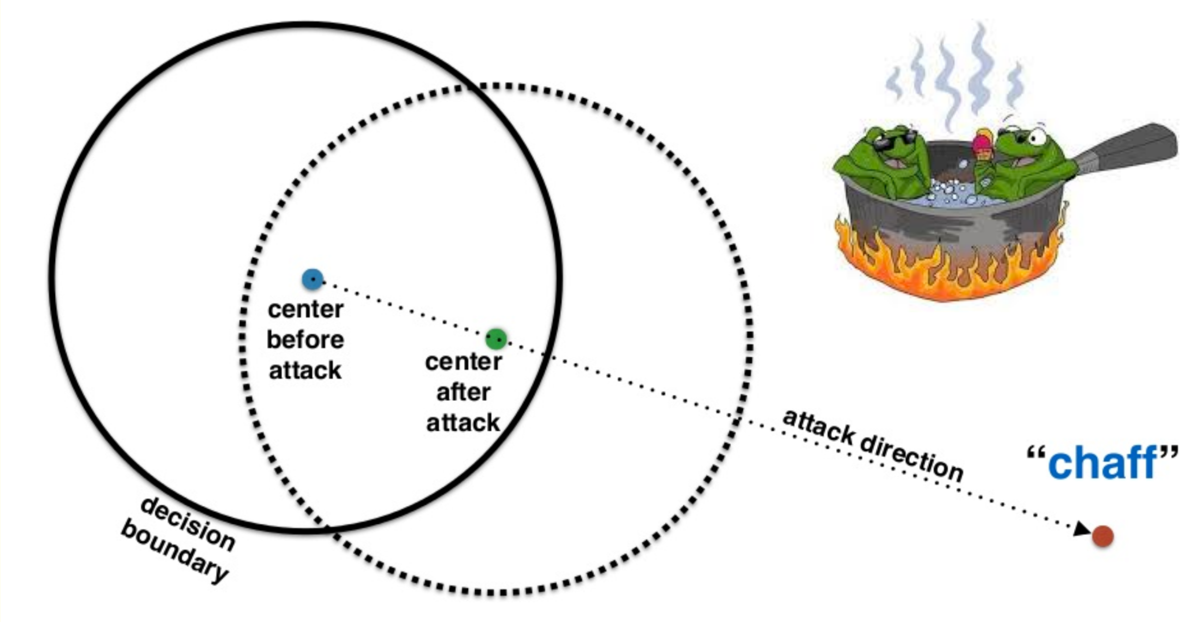
- Exploratory attacks: runtime
  - Find model vulnerabilities
  - Utilize invariant features



Gibbon Sample

[Szegedy et. al. 2014]

- Causative attacks: training time
  - Data Sanitization (e.g., RONI)
  - Robust Learning



[Rubinstein et. al. 2009]

Anomaly-based IDS: *Challenges*

- Problem: the identification of points which do not conform to an expected structure in a given dataset.

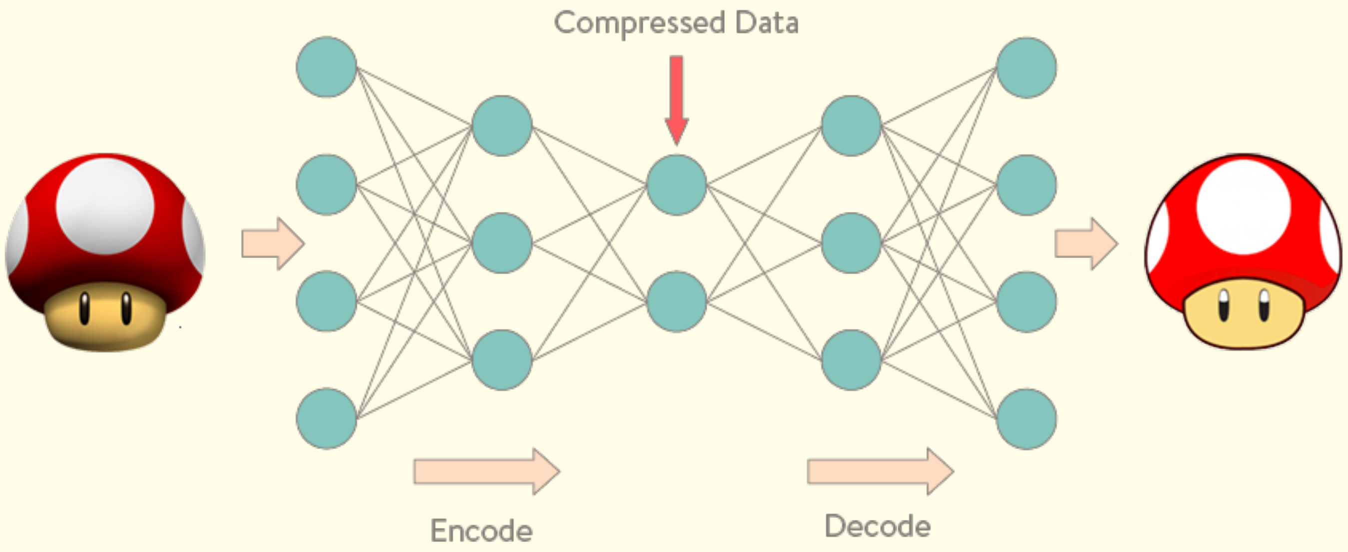
- E.g., anomaly-based IDS:
  - Build model(s) *M* explaining the expected behaviors (i.e., non-malicious) .
  - For a given point *x*, measure the likelihood of generation *p(x|M).*
  - Declare anomalousness based on the computed likelihood.

- Challenges:
  - What if "expected normal" changes over time (i.e., concept drift)?
  - Can adversary exploit the coping mechanism(s) for introducing malicious data points?
  - What are the affects of introduced contamination on the subsequent generated models?

Deep Autoencoders

- Is an *artificial neural network (ANN)*, where the purpose of output layer is to reconstruct the input of the network.

- For input x
  - encoding, i.e., *f(x)*
  - decoding, i.e., *g(f(x))*
  - optimize for loss function, i.e., *L(x, g(f(x)))*



- It exploits the idea that data concentrates around a low-dimensional manifold(s).
  - It will learn the structure of the manifold(s).

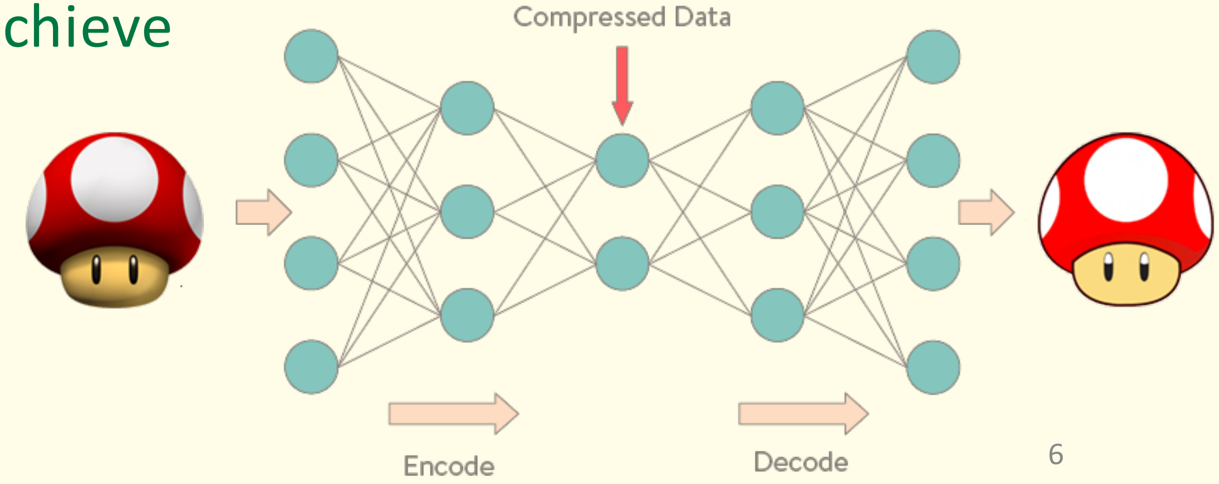
Deep Autoencoders for Anomaly Detection

- Deep autoencoders learn sophisticated manifolds thanks to cascaded layers of nonlinear computational units (i.e., neurons)
  - Once trained, the amount of incurred loss of reconstructed input can serve as measure of deviation of input X in respect to the expected dataset the deep autoencoder is representing.

$$L(x, g(f(x))) = \frac{1}{n} \sum (x_i - g(f(x_i)))^2$$

$$L(x, g(f(x))) > C$$

  - The threshold *C* is set empirically to achieve desired sensitivity and acceptable rate of false-positive.



Deep Autoencoders for Anomaly Detection:

the good, the bad, and the ugly

- General idea
  - Train a deep autoencoder on a non-malicious dataset.
  - Measure **how good** it can reconstruct non-malicious data points.
  - Empirically compute decision threshold value *C* based on desired rate of false-positive.
  - Measure **how bad** it reconstructs malicious data points.

- Our trained deep autoencoder specification
  - Categorical values are processed using *one-hot encoding*
  - *Two hidden layers of size 50 sigmoid neurons and one hidden layer of 10 sigmoid neurons.*
  - *Stochastic Gradient Descent*

The Experiments: *the dataset*

- NSL-KDD (Tavallaei *et. al. 2009*)
  - Resolved statistical flaws of the original KDD’99 intrusion detection dataset such as record redundancies.

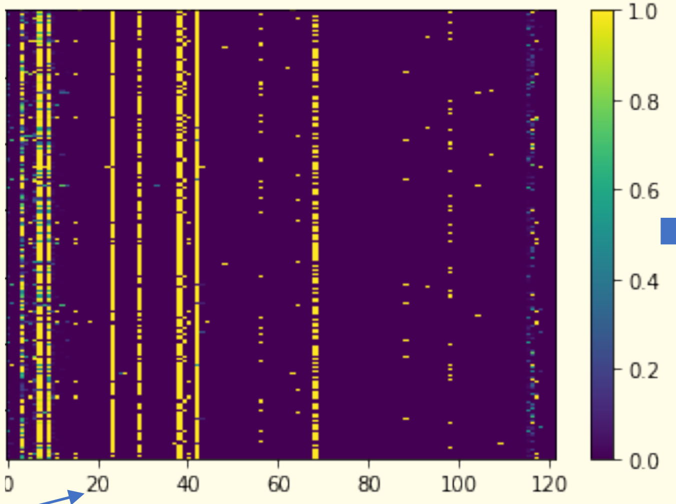
- Used *812,814* normal instances (i.e., non-malicious) to train the deep autoencoder.

- Used 29,378 attack instances and 47,911 normal instances (from test set) to evaluate the trained deep autoencoders.

The Experiments: *(1) intrusion detection*

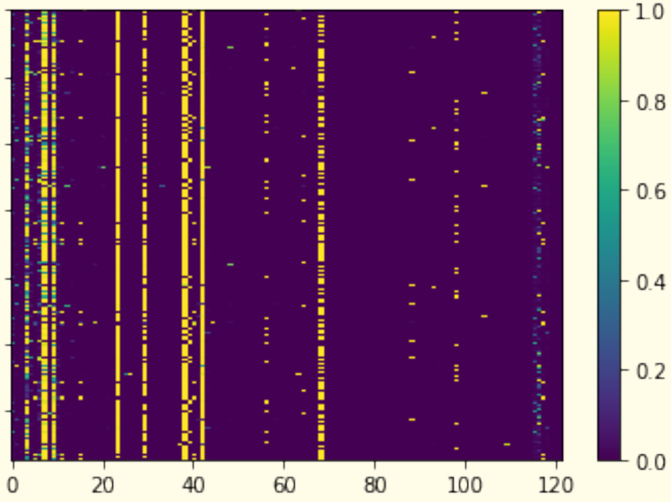
- How well the deep autoencoder is capable of reconstructing normal instances?
  - *[In the heatmaps bellow, each row is representing a normal instance (vertical axis) and each column is representing corresponding feature value (horizontal axis).]*

Instance Index



Feature Index

Original Normal Instance Heatmap

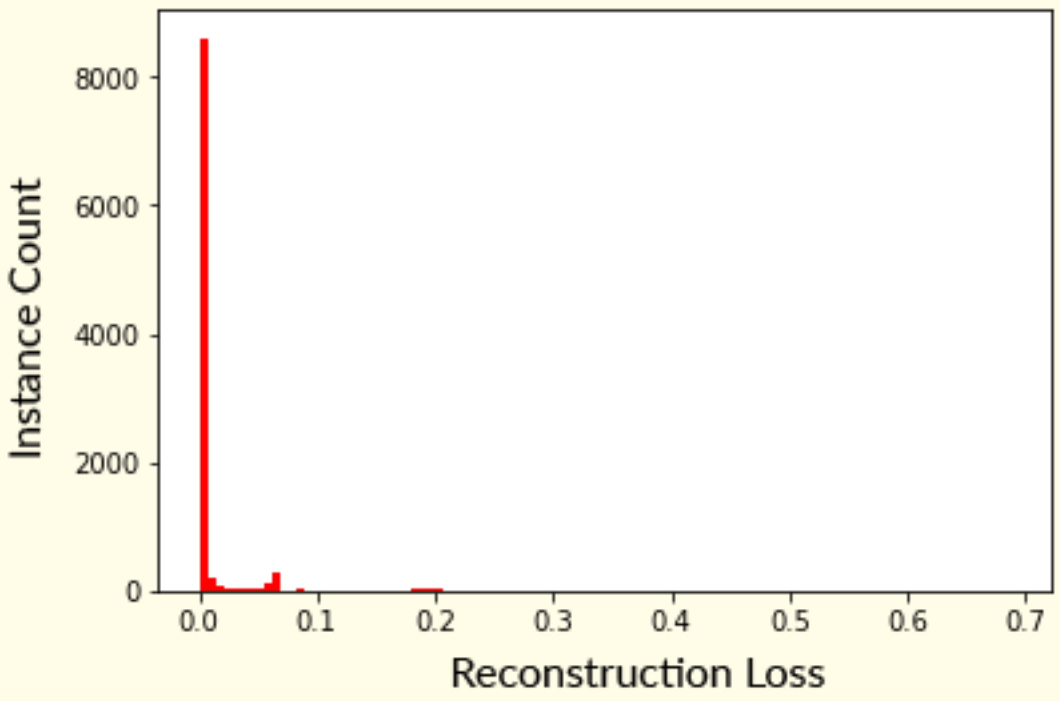


Reconstructed Normal Instance Heatmap

The Experiments: *(1) intrusion detection - cont'd*

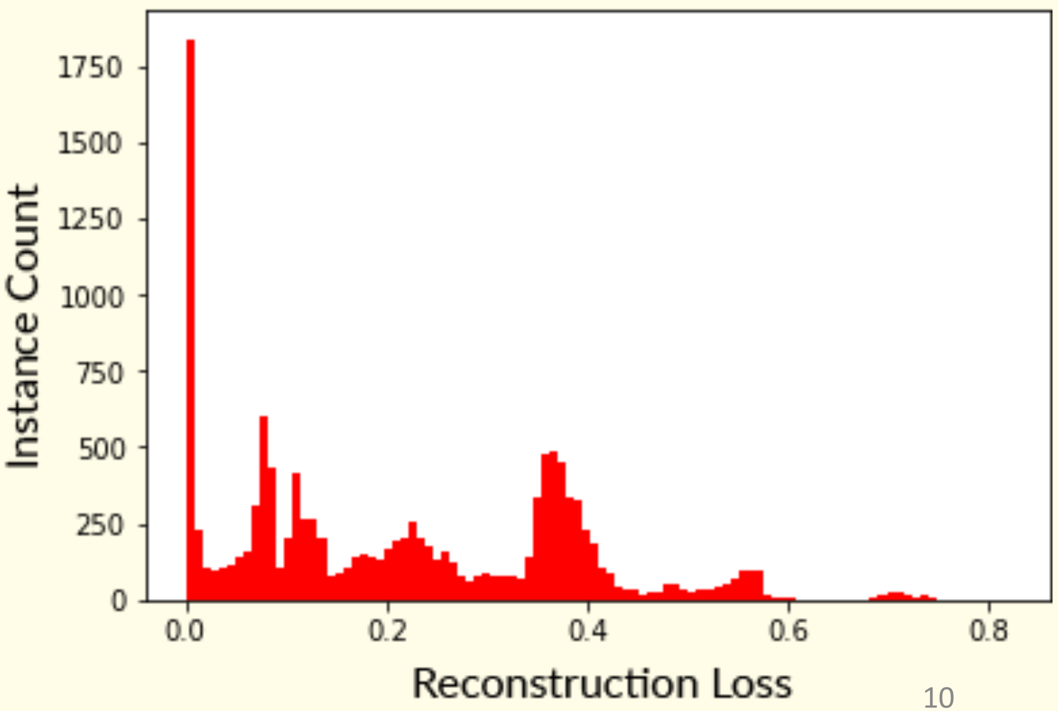
- How bad the deep autoencoder constructed malicious instances?
  - *[In the heatmaps bellow, each row is representing a normal instance (vertical axis) and each column is representing corresponding feature value (horizontal axis).]*

Non Malicious Instances



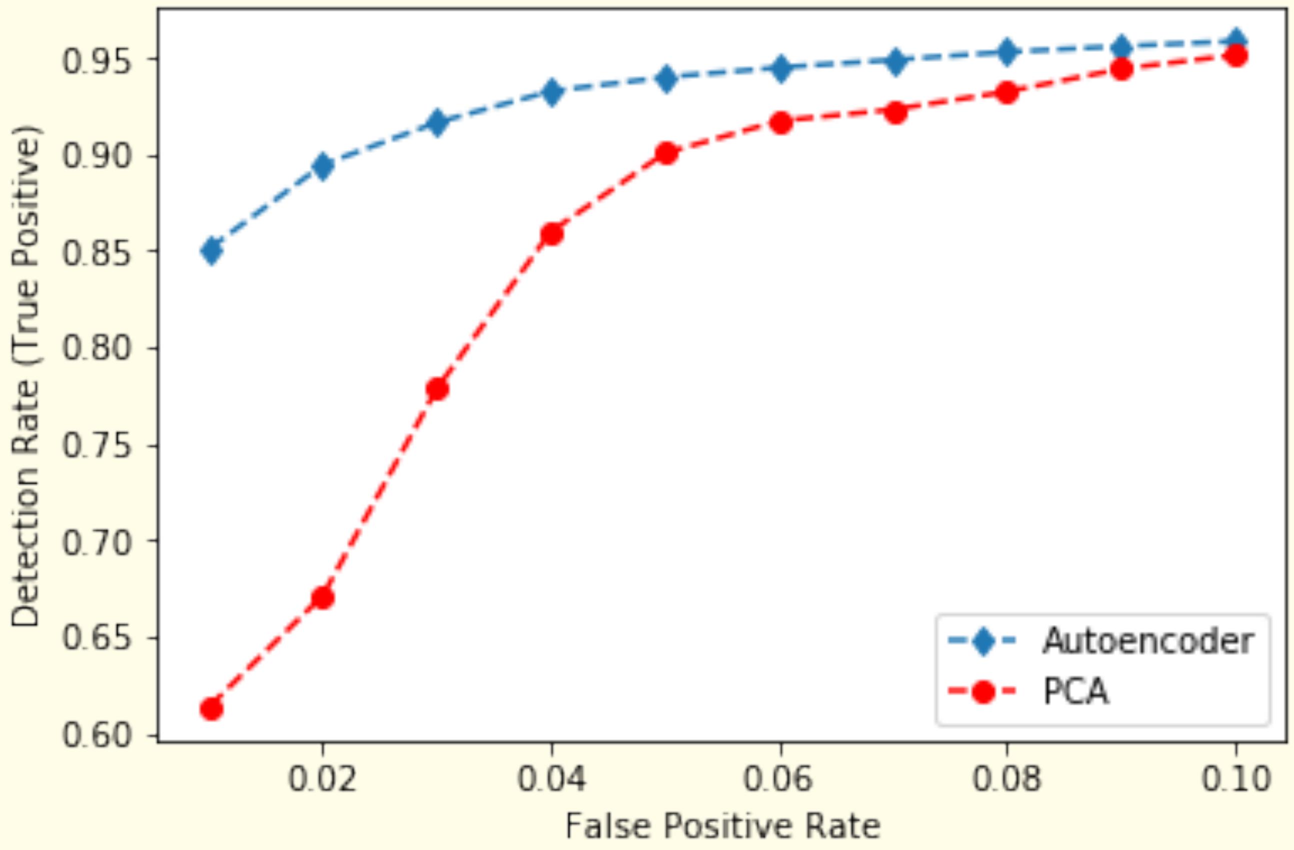
VS

Malicious Instances



The Experiments: *(1) intrusion detection - cont'd*

- Compared anomaly detection performance of the deep autoencoder with *Principle Component Classifier* (PCC) by *Shyu et. al. 2003*



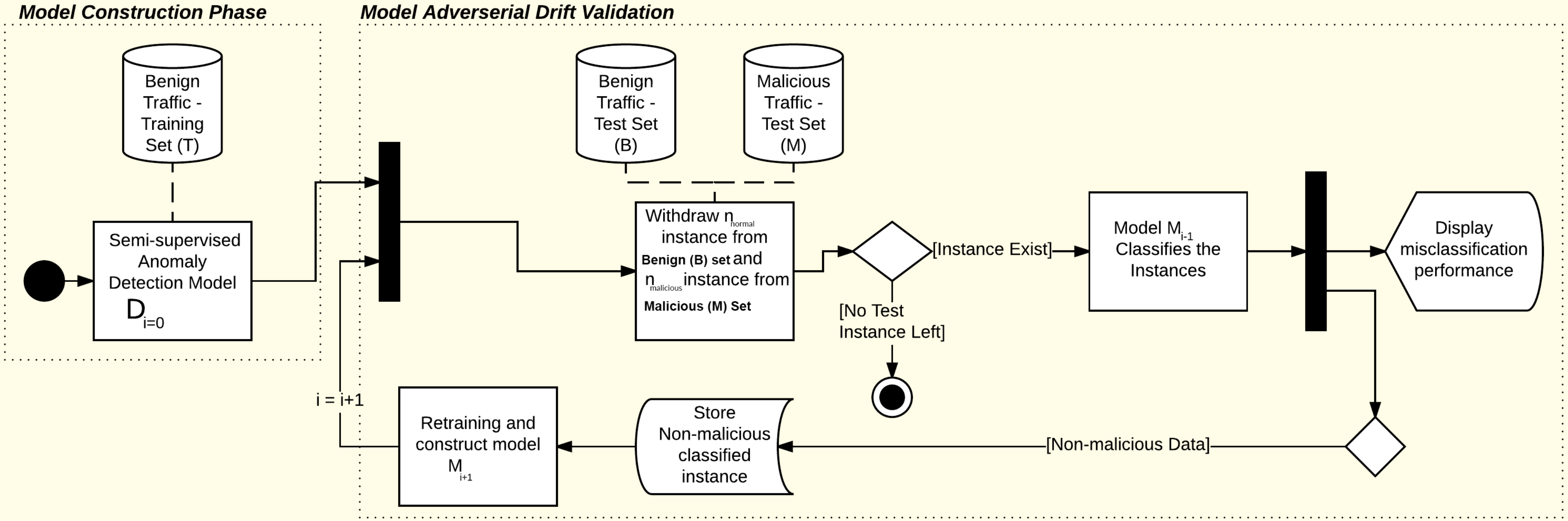
But, how about robustness?

- ANTIDOTE (Rubinstein *et. al.* 2009) proposed set of invariant feature transformations to make PCC more robust against training data noise.
  - Principle Component Analysis, though easy to implement and scale, is extremely sensitive to presence of noise.
  - PCC's boundary decisions can manipulated using adversarial contaminations.

- How deep autoencoders perform under noise and/or adversarial contaminations?
  - Recall, the non-malicious data distribution(s) that are used for model training is <u>not stationary.</u>

Our Proposed Framework: *for evaluating detection models under adversarial influence*

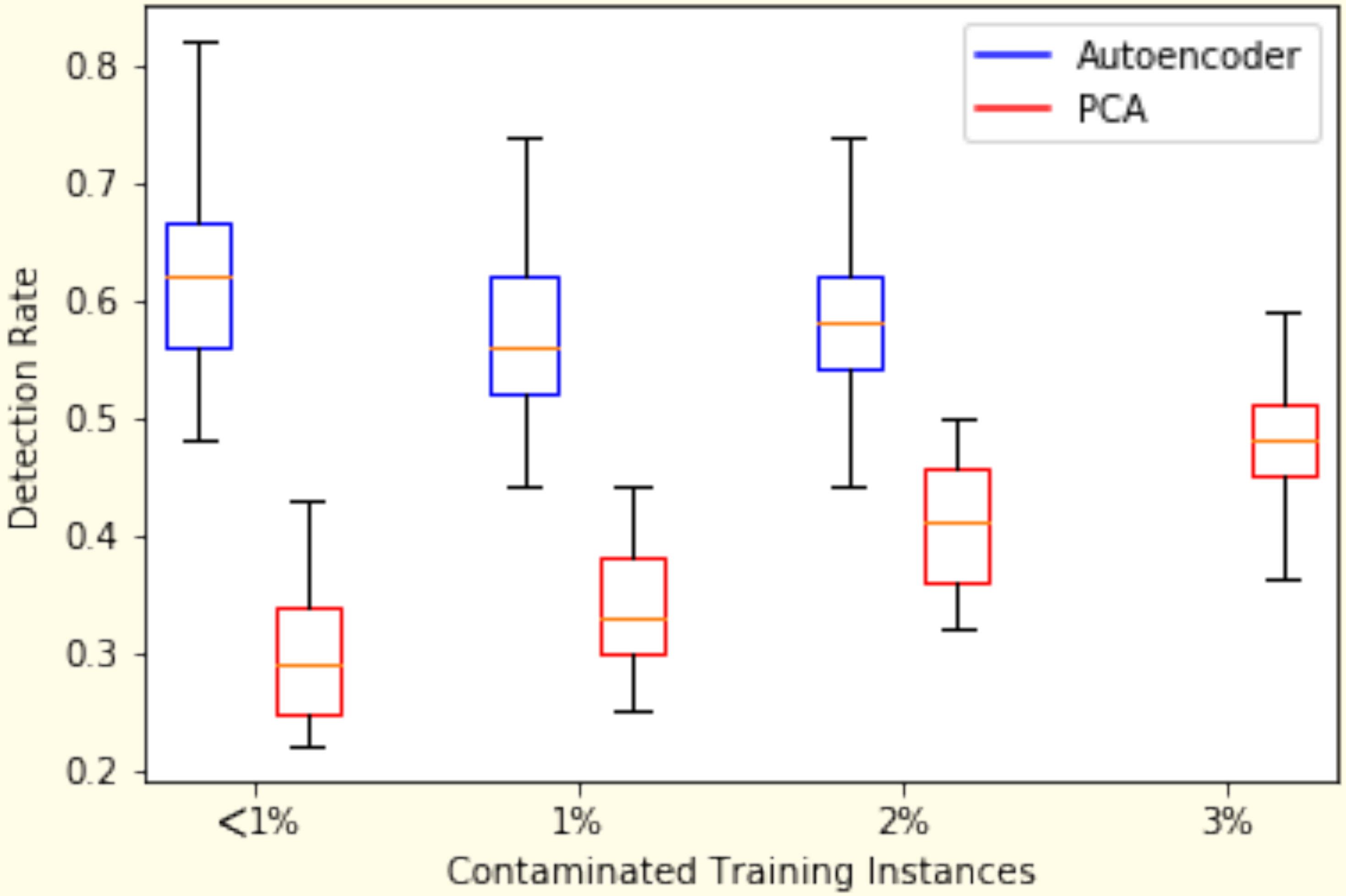
- Our goal is to measure robustness of adaptive (i.e., online) anomaly-based IDS that update in an unsupervised fashion
  - Assumption: use newly classified data points with high confidence to construct retraining dataset
  - Normal data drift slowly and gradually



Our Proposed Framework – cont'd

- The main idea:
  1. Initially train a classifier for detecting malicious activities.
  2. Construct some test dataset containing both malicious and normal data points.
  3. Let the trained classifier to classify data points and capture classification performance.
  4. Use an arbitrary selection function to choose recently classified data points for enhancing the training dataset.
     - False-positive classified data points will result in loosing valuable new data points for enhancing the training dataset.
     - False-negative classified data points will result in contaminating existing training dataset.
  5. Retraining the classifier using enhanced dataset.
  6. Repeat and track recorded detection performance!

Robustness of the Deep Autoencoder vs PCC



Discussions and Future work

- Deep Autoencoders maintain a more stable sensitivity in the light of contaminations, and it suffers less in respect to other subspace analysis methods such as PCA.
- Retraining deep autoencoders can be done by running new training examples through the existing network without start the training from scratch -> Online Learning by default
- Deep Autoencoders can be used to estimate the underlying probability distributions explaining the training dataset.
  - It can be used to compute inference – a different notion to measure anamoly.
  - Can be used to generate examples – adversarial examples to evaluate arbitrary detection models.