

AMERICAN UNIVERSITY OF BEIRUT

SECURITY OF ABSTRACTION BASED  
NOVELTY DETECTION IN DEEP  
LEARNING

by

SARA IBRAHIM HAJJ IBRAHIM

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Science  
to the Department of Computer Science  
of the Faculty of Arts and Sciences  
at the American University of Beirut

Beirut, Lebanon  
June 2021

AMERICAN UNIVERSITY OF BEIRUT

SECURITY OF ABSTRACTION BASED  
NOVELTY DETECTION IN DEEP  
LEARNING

by

SARA IBRAHIM HAJJ IBRAHIM

Approved by:

---

Dr. Mohamed Nassar, Assistant Professor

Computer Science

Advisor



---

Dr. Shady Elbassuoni, Associate Professor

Computer Science

Member of Committee

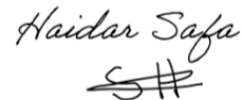


---

Dr. Haidar Safa, Professor

Computer Science

Member of Committee



Date of thesis defense: June 14, 2021



# Acknowledgements

First, I would like to thank AUB for this wonderful journey and experience. Thank you for facilitating such a good learning environment especially during a pandemic. I'm very thankful for all what you offer for your students.

I'm also deeply grateful to all people who continued to support me earlier in my journey at AUB to my completion of my thesis.

First, I would like to express my deepest appreciation to my advisor Dr. Mohamed Nassar, it wouldn't have been possible without his constant supervision and guidance. I would also like to extend my sincere thanks to my thesis committee members, Dr. Shady and Dr. Haidar for their time and extremely valuable advice.

To my family, my parents and siblings, my life mentors: I'm sorry you put up with me being missing many out-gatherings and events. I want to thank you for all your sacrifices and beliefs in me. Your constant love and support went above and beyond to help me reach my goals.

To my second family, my friends, who we've shared fun and misery together, thank you for incorporating humor and studies; whether we took long breaks in the middle of our study times, went out or held hours of talk.

Finally, I would like to thank myself for always being a fighter for my goals and never quitting. Afterall, I attribute my success, strength, accomplishment, good fortune, consistence and my outcomes all to God.

# An Abstract of the Thesis of

Sara Hajj Ibrahim for Master of Science  
Major: Computer Science

Title: Security of Abstraction based Novelty Detection in Deep Learning

Deep learning is a type of machine learning that adapts a deep hierarchy of concepts. Deep learning classifiers link the most basic version of concepts at the input layer to the most abstract version of concepts at the output layer, also known as a class or label. However, once trained over a finite set of classes, a deep learning model does not have the power to say that a given input does not belong to any of the classes and simply cannot be linked. Correctly invalidating the prediction of unrelated classes is a challenging problem that has been tackled in many ways in the literature.

Novelty detection gives deep learning the ability to output "do not know" for novel/unseen classes. Still, no attention has been given to the security aspects of novelty detection. In this thesis, we study the case of abstraction-based novelty detection in deep learning in particular. We show that abstraction-based novelty detection is not robust against adversarial attacks.

We formulate three types of attacks against novelty detection: (1) passing a valid sample as invalid, (2) passing an invalid sample as valid, and (3) passing an adversarial sample as valid. We experiment different optimisers for solving our formulated attacks (1 & 2) on multiple neural network architectures. For attack (3), we show the feasibility of an adversarial sample that fools the deep learning classifier to output a wrong class. We follow existing adversarial attacks by our proposed optimisation attack to bypass the novelty detection monitoring at the same time.

In other words, we show that we can break the security of novelty detection. We call for further research on novelty detection from a defender's point of view. We

adapt a suitable defense mechanism against such attacks and assess its performance.

The thesis suggests that more attention could be paid in novelty detection systems to make them more secure against attacks. Especially in critical-decision making systems that are based on artificial intelligence and machine learning, for example self-driving cars, automated medicine or cybersecurity. To our knowledge, our work is the first to address the security limit of novelty detection in deep learning.

# Contents

<b>Acknowledgements</b>	v
<b>Abstract</b>	vi
<b>1 Introduction</b>	1
1.1 Deep learning . . . . .	1
1.2 Challenges in deep learning . . . . .	1
1.3 Novelty detection . . . . .	2
1.4 Security of novelty detection . . . . .	2
1.5 Thesis contributions . . . . .	2
1.6 Broader impacts . . . . .	4
1.7 Roadmap of thesis . . . . .	5
<b>2 Literature Review</b>	6
2.1 Outlier vs. Anomaly detection . . . . .	6
2.2 Novelty detection . . . . .	8
2.3 Approaches addressing novelty detection . . . . .	9
2.4 Security risk . . . . .	18

<b>3 Methodology</b>	<b>20</b>
3.1 Optimisation based attacks	21
3.2 Adversarial attacks against neural networks	26
3.3 Defense mechanism against attacks	29
<b>4 Experimental setup</b>	<b>31</b>
4.1 Attacks experimental setup	31
4.2 Defense experimental setup	35
<b>5 Optimisation solvers experiment</b>	<b>36</b>
5.1 Attacks on network 1	36
5.1.1 Attack 1 on network 1	36
5.1.2 Attack 2 on network 1	38
5.2 Attacks on network 2	40
5.2.1 Attack 1 on network 2	40
5.2.2 Attack 2 on network 2	42
5.3 Attacks on MNIST classifier	43
5.3.1 Attack 1 on MNIST classifier	43
5.3.2 Attack 2 on MNIST classifier	47
5.4 Discussion	49
<b>6 Adversarial attacks experiment</b>	<b>54</b>
6.1 Tuning number of clusters	57
6.2 Tuning tolerance factor	58
6.3 Optimisation attack again!	61
<b>7 Defense mechanism against attacks</b>	<b>65</b>



<b>7.1</b>	<b>Effect of denoising auto-encoders against optimisation attacks</b>	<b>. . 66</b>
<b>7.2</b>	<b>Effect of denoising auto-encoders against neural network attacks</b>	<b>. 68</b>
<b>8</b>	<b>Conclusion</b>	<b>71</b>
<b>A</b>	<b>Research Article</b>	<b>73</b>
<b>B</b>	<b>Abbreviations</b>	<b>84</b>

# List of Figures

2.1	Outlier detection definition	7
2.2	Anomaly detection definition	7
2.3	Novelty detection definition	8
2.4	An example of outlier detection system with a statistical and reconstruction based method	12
2.5	An example of anomaly detection system with a distance and probabilistic based method	14
2.6	An example of novelty detection system with a distance and kernel-space based approach	15
2.7	An example of novelty detection system with an abstraction based approach	17
3.1	Optimisation attack generalization	23
3.2	Denoising auto-encoders architecture	30
4.1	Network 1 architecture	32
4.2	Decision Boundary plot for network 1	33
4.3	Network 2 architecture	34
4.4	Decision Boundary plot for network 2	35

5.1	Success rate of different optimisation based methods for attack 1	
	on network 1	37
5.2	Decision boundary plot of attack 1 on network 1	38
5.3	Success rate of different optimisation based methods for attack 2	
	on network 1	39
5.4	Decision boundary plot of attack 2 on network 1	39
5.5	Success rate of different optimisation based methods for attack 1	
	on network 2	40
5.6	Decision boundary plot of attack 1 on network 2	41
5.7	Success rate of different optimisation based methods for attack 2	
	on network 2	42
5.8	Decision boundary plot of attack 2 on network 2	43
5.9	Success rate of different optimisation based methods for attack 1	
	on MNIST classifier	44
5.10	2d projection of attack 1 attempts on MNIST classifier: 2 classes	
	involved	45
5.11	2d projection of attack 1 attempts on MNIST classifier: 3 classes	
	involved	46
5.12	Success rate of different optimisation based methods for attack 2	
	on MNIST classifier	47
5.13	2d projection of attack 2 attempts on MNIST classifier: 2 classes	
	involved	48
5.14	2d projection of attack 2 attempts on MNIST classifier: 3 classes	
	involved	49

5.15	Success rate of different optimisation based methods for the four proposed attacks on different network architectures.	50
5.16	Attack 1 adversarial examples obtained by SHGO method	52
5.17	Attack 2 adversarial examples obtained by SHGO method	53
6.1	Experiment measuring the number of successful and valid attack attempts using neural network attacks	55
6.2	Experiment measuring the number of successful and valid attack attempts using neural network attacks while tuning different number of clusters	57
6.3	2d projection of FGSM attack attempts while tuning different number of clusters	58
6.4	Experiment measuring the number of successful and valid attack attempts using neural network attacks while tuning different tolerance	59
6.5	2d projection of FGSM attack attempts while tuning different tolerance factor	60
6.6	Fooling the classifier and the monitor together: FGSM adversarial samples acceptance after optimisation	61
6.7	Fooling the classifier and the monitor together: $L_2$ FastGradient adversarial samples acceptance after optimisation	62
6.8	Image samples after dual attack (FGSM + attack 2)	63
6.9	Image samples after dual attack ( $L_2$ FastGradient + attack 2)	64
7.1	Success rate of SHGO optimisation attack after defense	66
7.2	Defending an image sample against optimisation attacks	67

7.3	Success rate of neural network adversarial samples after defense	. 68
7.4	Defending an image sample against neural network attacks	. . . 70

# List of Tables

3.1 Foolbox neural network adversarial attacks and description . . . . 28

6.1 Foolbox neural network adversarial attack result . . . . . 56

A.1 Reviewers comments . . . . . 76

# Chapter 1

## Introduction

### 1.1 Deep learning

Machine learning algorithms are excellent at analyzing data and finding interesting patterns. However, they give up to the so-called dimensionality curse. Neural networks are wildly successful today [1]. Neural networks bypass the traditional machine learning algorithms in most learning tasks and can help combat the problem of this "curse" [2].

### 1.2 Challenges in deep learning

While deep learning yields remarkable results in the field of raw data representation and classification, neural networks are sub-optimal when: (1) explaining decisions or (2) recognizing a novel class of input. Deep learning does not really give clear insights about the process of its decision making. Moreover, deep neural networks never say "I don't know", they can be just less or more confident

about an outcome or a decision. In supervised learning, a deep learning classifier is trained over a set of classes. The classifier always outputs that a given sample is more likely to belong to one of the classes, never to any of them.

### **1.3 Novelty detection**

The goal of **novelty detection** is to decide whether a given input sample really belongs to one of the classes or is completely drawn from a statistical distribution that was unseen during the training phase. Novelty detection can output: "I don't know the class of this sample, it is not among the classes supported by this classifier" or "I have seen this class before and the classifier decision is valid". In other words, the novelty detection decision may invalidate the neural network decision.

### **1.4 Security of novelty detection**

Though novelty detection is receiving a good deal of coverage in the literature, its performance is usually measured by its accuracy on detecting novel inputs under a benign threat model. However, the security of novelty detection however seems not to have received much attention in literature.

### **1.5 Thesis contributions**

The subject of this thesis is to undertake a systematic study of security aspects of novelty detection. We test the robustness of novelty detection systems against



adversarial attacks. We propose attacks that do slight modifications of the testing input samples.

1. We design 2 optimisation based attacks capable of generating adversarial samples: (1) passing a valid testing sample as invalid and (2) passing an invalid testing sample as valid.

We test and compare different optimisers for solving the attack formulations and conclude that SHGO optimiser is the best solver. We experiment with two basic neural networks as well as a convolution neural network for MNIST classification. We conduct an extensive set of experiments to show that the generated attack samples can be perceived as normal samples by a human observer.

2. We consider another type of attack that passes an adversarial testing sample as valid. An adversarial sample is a valid sample of some class A that has been modified to fool the neural network into deciding it as a class other than A. One may assume that adversarial samples should be detected as novelties. We show that this is not always true, (1) adversarial samples may be detected as not novel and (2) we can circumvent the decision of the novelty detector to make adversarial samples appear as not novel.

We show the feasibility of crafting adversarial samples that fool the deep learning classifier and bypass the novelty detection monitoring at the same time. Such samples are constructed in two steps: (1) a classical adversarial attack followed by (2) our proposed optimisation attack suggested in 1.

In other words, we show that novelty detection can be hackable. We demonstrate

that novelty detection itself ends up as an attack surface. Finally, we propose to adapt denoising auto-encoders as an efficient defense mechanism against such attacks and assess its performance.

This thesis is an attention call for raising awareness of security aspects in artificial intelligence and machine learning. Security design is required when integrating artificial intelligence into critical decision making systems such as self-driving cars, automated medicine or cyber security. Our work is the first to address the security of novelty detection in deep learning.

## **1.6 Broader impacts**

We estimate that novelty detection will play a significant role in monitoring neural networks and discovering new classes unseen during training time. Of course, attackers will not cooperate with us in this task. Attackers may take steps to evade novelty detection systems and avoid notice. The question of security is extremely crucial in critical systems that involve healthcare assessments or hospital services, IT security systems or tech companies, self driving cars, etc., systems where the error cost is very expensive. We expect attackers to come up with several strategies to make these systems less effective. Hence, the security of novelty detection systems is a useful and important topic. We should make it difficult for attackers to cause harm and take control using efficient defense mechanisms. In this sense, the very success of a novelty detection approach depends on its security against attacks.

## 1.7 Roadmap of thesis

In chapter 2, we differentiate novelty detection as a new domain compared to classical anomaly detection and outlier detection domains. Moreover, we survey different examples of anomaly, novelty and outlier detection and address approaches that mainly convey novelty detection. Then, in chapter 3, we try to explore the limits of novelty detection security. We conduct several experiments of (1) optimisation based attacks and (2) adversarial neural network attacks. In chapter 5 and 6, we report the results and findings of our attack experiments. A good idea is then to adapt a defense mechanism in response to attacks. Next, we show that using a denoising auto-encoder hinders the effect of attacks in chapter 7. We finally conclude the thesis and sketch future work.

# Chapter 2

## Literature Review

In this chapter, we distinguish between three interrelated and very close concepts, namely anomaly detection, outlier detection and novelty detection. These terms are sometimes interchangeably used in literature, but we suggest that they mean different things. We need to be more precise about what is novelty detection and how it differs from outlier and anomaly detection. So we highlight the difference between outlier, anomaly and novelty detection early before we proceed.

### 2.1 Outlier vs. Anomaly detection

An outlier is an "Observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism" [3]. Outlier detection is the process of detecting an observation that significantly deviates from the majority of data. Unsupervised algorithms extract statistical information indicating how unlikely a certain observation is, for example, finding a point deviating far from the statistical means of other points as illustrated in figure 2.1.

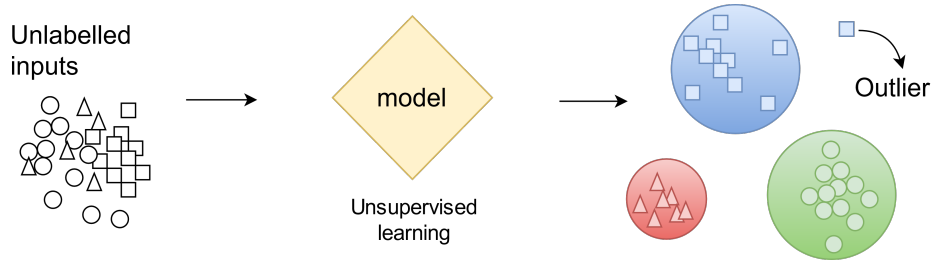


Figure 2.1: Outlier detection definition

An anomaly however is a special case of outliers which is usually tied to special information or reasons [4]. Anomalies indicate significant and rare events that may prompt critical actions in a wide range of application domains [5]. Such triggers are considered malicious and should be directly rejected by the system. Anomaly detection may require labeled data and employ supervised algorithms as illustrated in figure 2.2. As an example, we consider the problem of malware/benign classification as a form of anomaly detection.

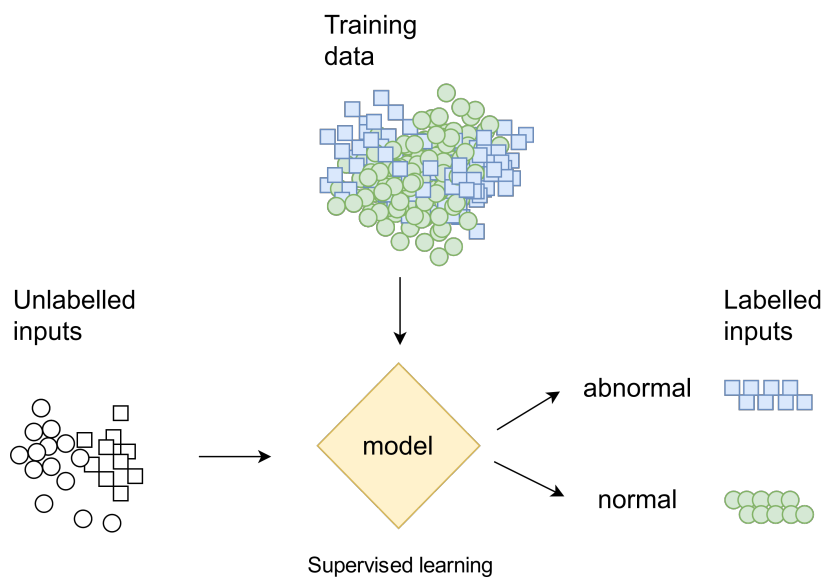


Figure 2.2: Anomaly detection definition

## 2.2 Novelty detection

Novelty detection is the process of identifying inputs that belong to unknown classes not seen during training time. Assume we have  $c$  classes during training time,  $c + u$  classes appear at testing time. The goal of novelty detection is to invalidate the output of the classification when samples from the  $u$  classes are presented. Novelty detection is different than the previously described anomaly and outlier detection for two main reasons: (1) training data have labels, and (2) the learner itself is an input to the detector algorithm.

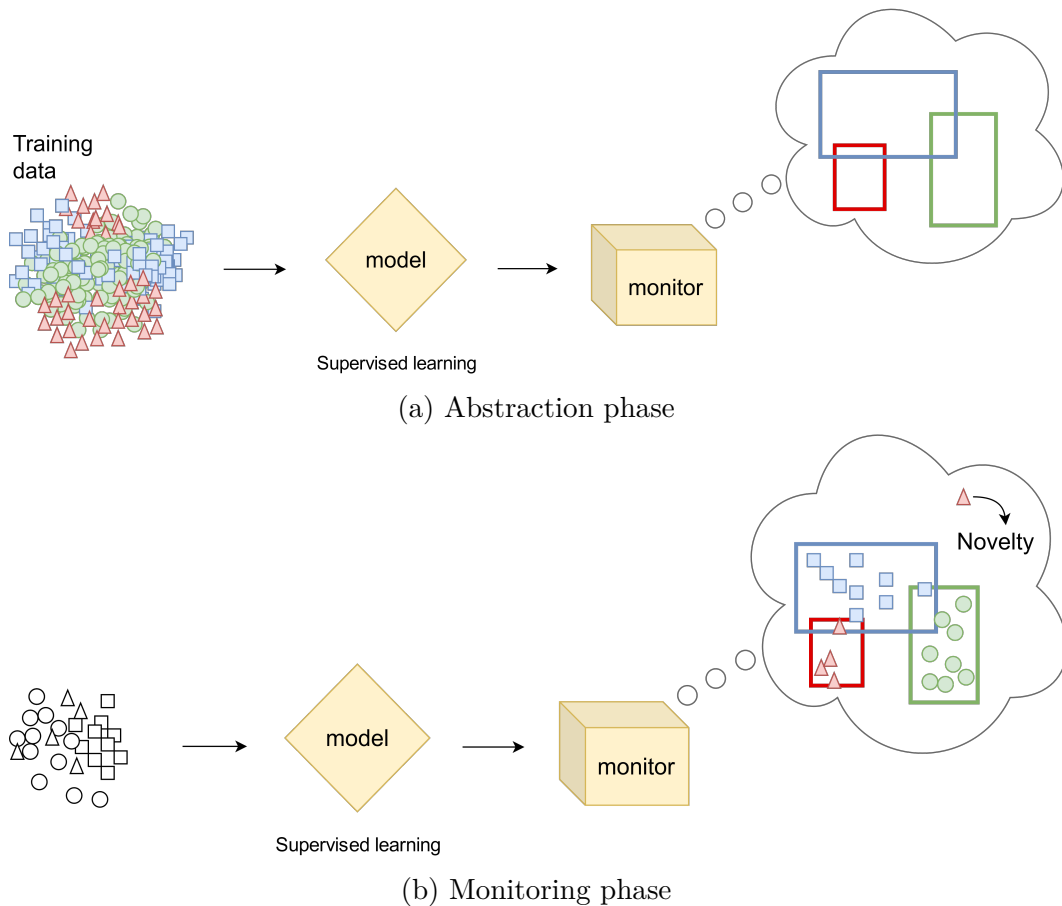


Figure 2.3: Novelty detection definition

Novelty detection can be achieved in white-box mode by taking the model ob-

tained after training and building a monitor on top of it. The monitor fingerprints the behaviour of the model when training data are presented. For the special case of deep learning, such a monitor can register the values of hidden nodes given by forward-propagating the training samples. The monitored values are abstracted into statistical constructs. Later, outlier detection flags inputs having fingerprints that largely deviate from these constructs. In other words, this approach transforms the novelty detection problem into an outlier detection problem by projecting the data into a different hyper-dimensional space. This projection is parameterized by the neural network model itself. Different types of abstractions are proposed in [6] before evaluating box abstractions in particular. Figure 2.3 shows an example of a box abstraction-based novelty detection system.

## 2.3 Approaches addressing novelty detection

We categorise the main approaches that address novelty detection. These methods also apply for classical anomaly and outlier detection. For our study, we group approaches into distance based, statistical based, auto-encoding & reconstruction based, bayesian and abstraction based approaches.

**Distance based methods** These methods compute novelty values or confidence scores based on distance metric functions. In [7], data is first embedded as derived from the penultimate layer of the neural network. The confidence score is based on the estimation of local density. Local density at a point is estimated based on the Euclidean distance in the embedded space between the point and its  $k$  nearest neighbors in the training set. A similar approach based on learning a local model around a test sample is proposed on [8] for

Multi-class novelty detection tasks in image recognition problems.

**Statistical based methods** Novelty is caused by differences in data distributions at training and prediction time. Some of these methods require sampling the distribution at run-time or an online adaptation of classifiers. In [9] the underlying structure of the inlier distribution of the training data is captured. The novelty is detected by the means of a hypothesis test or by computing a novelty probability value.

**Auto-Encoding and reconstruction based methods** One way to proceed is to train a deep encoder-decoder network that outputs a reconstruction error for each sample. The error is used to either compute a novelty score or to train a one-class classifier. For example [10] introduces an unsupervised model for novelty detection based on Deep Gaussian Process Auto-Encoders (DGP-AE). The proposed auto-encoder is trained by approximating the DGP layers using random feature expansions, and by performing stochastic variational inference on the resulting approximate model. Their work can be categorized under anomaly detection in our terminology.

**Bayesian methods** These methods use bayesian formalism to detect anomalies and new classes in addition to classification [11]. The basic idea is to add a "dummy" class at the root node. The class is considered under-represented in the training set. The classifier gives a strong a posterior of being "dummy" for unseen instances.

**Abstraction based methods** These methods consider a finite set of vectors  $\mathbf{X}$ , and construct a set  $\mathbf{Y}$  that generalizes  $\mathbf{X}$  to infinitely many elements and has



a simple representation that is easy to manipulate and answer queries for. Examples of these methods are ball-abstraction such as one-class support vector machines, one-class neural networks [12] and box-abstraction [6] that we will describe later.

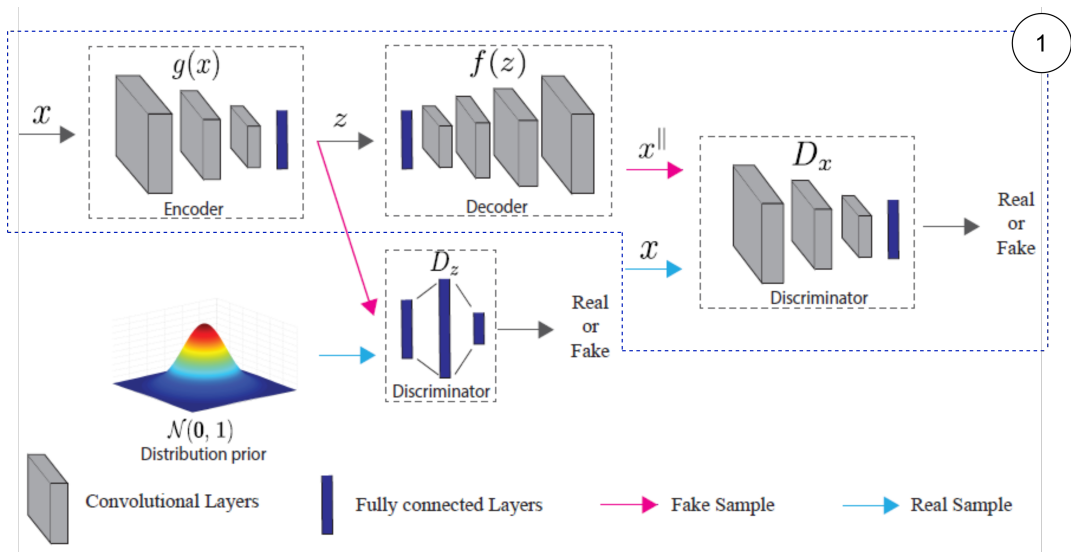
Next, we show examples of different approaches supporting some categories of the preceding methods.

## **An example of statistical and auto-encoding/ reconstruction based method**

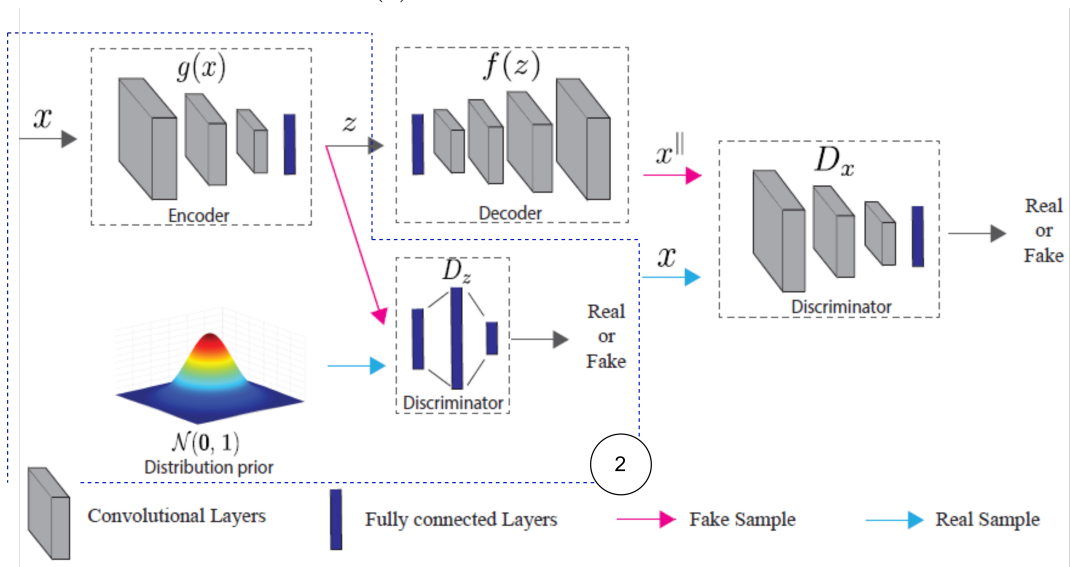
[9] trains an auto-encoder network to derive a linearized manifold representation of data. The manifold representation helps compute a novelty probability that represents how likely it is that a sample was generated by the inlier distribution. This is why we consider it as a statistical based method at the same time. We train an auto-encoder using two steps:

1. Reconstruction step: Auto-encoders learn a representation over data by training to copy inputs to outputs. First an encoder transforms an input image from high into lower dimensional space, known as latent space  $z$ . Then a decoder defining a surface manifold  $M$  reconstructs back the original image according to information already learned. Once learned, a discriminator is then able to distinguish reconstructed inputs as fake and original samples as real (see figure 2.4 - a).
2. Generative step: Auto-encoders impose that a normal distribution and sam-

ples generated by  $z$  be as close to each other. We don't want passed inputs to be deviating far from the normal data. Once learned, a discriminator is able distinguish samples generated from  $z$  as fake and inputs from normal distribution as real (see figure 2.4 - b).



(a) Reconstruction Phase



(b) Generative phase

Figure 2.4: An example of outlier detection system with a statistical and reconstruction based method

Once the training phase ends, a manifold  $M$  gets learned. During testing phase, we project testing samples into the latent space  $z$  and find local coordinates " $w$ ", the new coordinates in latent space. At this step, we compute the novelty score using the probability density function " $p_w(w)$ " of  $w$  coordinates, the multiplication of  $p_w(w^{\parallel})$  and  $p_w(w^{\perp})$ , the probability density functions of  $w$  coordinates parallel and orthogonal to the tangent space  $T$  in manifold. Once we compute  $p_w(w)$ , we say that the input sample acts as an inlier or an outlier over a certain threshold. We refer this approach as an example of outlier detection, as it doesn't require labels and spots outliers deviating from the normal distribution.

## An example of a distance and probabilistic based method

Another example tries to detect anomalies and well known attacks such as DOS, Probe, U2R and R2L [13]. It involves the following steps: (a) data pre-processing and feature reduction, and a two tier classifier: (b) Naive Bayes, defining the probabilistic based characteristic, and (c) KNN-CF, making our algorithm distance based. Figure 2.5 summarizes these steps.

- (a) Upon passing the training inputs, and for better decision making, we apply some transformation to the data: (1) convert categorical feature inputs into numerical, (2) normalize data values and (3) perform feature reduction.
- (b) Next, a Naive Bayes model is used to classify the data obtained in (a) as anomalous or not. The Naive Bayes is known as a classification model that calculates the probability that an input  $\mathbf{x}$  belongs to particular class  $c$  [14].

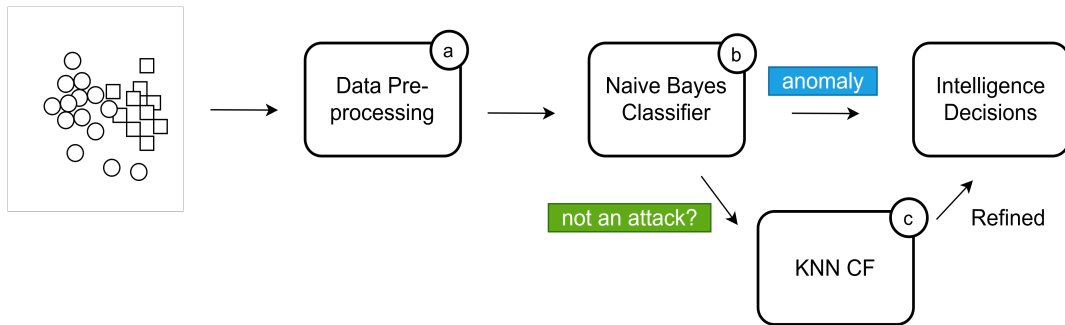


Figure 2.5: An example of anomaly detection system with a distance and probabilistic based method

- (c) Naive Bayes may lead sometimes to wrong decisions, so we pass normal inputs again into a KNN classifier, the certainty factor version. The KNN-CF classifier is same as the traditional KNN except we use the certainty factor to balance the data.

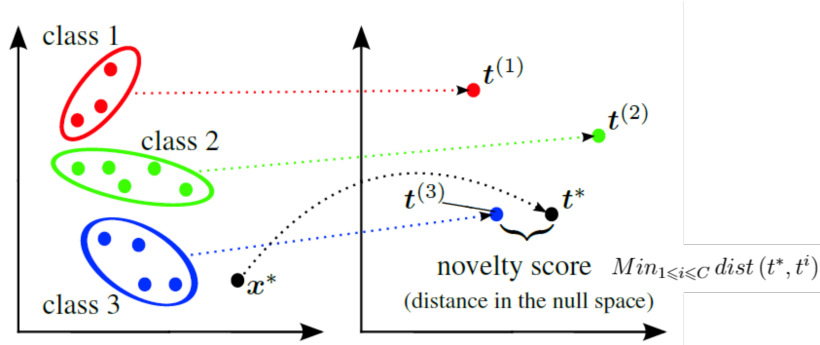
This approach falls in context of anomaly detection and we stem as an example of [2.2](#). It is a binary classification problem where we classify events as either normal or abnormal.

## An example of distance and kernel-space based method

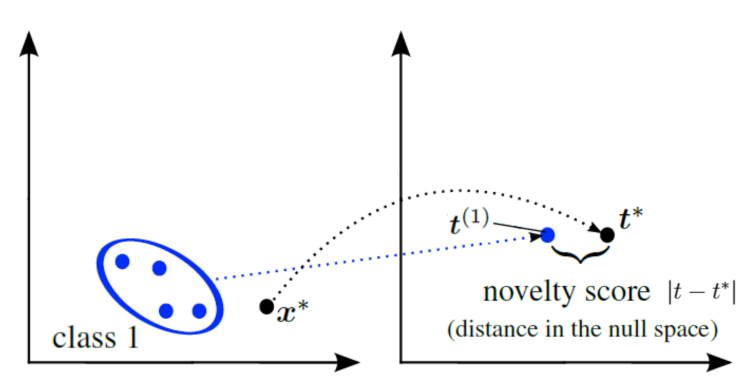
We propose in [\[8\]](#) an approach that recognizes a sample as novel if it is far away from its nearest neighbors in the training set. This approach tries to map inputs into null-space to compute its novelty score. We define null space as the space that consists of vectors which under some linear transformation are mapped onto zero [\[15\]](#).

Given training data with  $C$  classes, we map all given samples belonging to a class

$c^i$  into a single point in null subspace. So that we end up with a single point for each class  $c^i$  in null space, called the target point  $\mathbf{t}^i$ . We use one-class novelty detection if all neighbors in training set belong to the same class, otherwise we end up using multi-class novelty detector.



(a) Multi-classification



(b) One-classification

Figure 2.6: An example of novelty detection system with a distance and kernel-space based approach

- (a) In multi-classification (figure 2.6 - a), we test whether an input sample belongs to a certain class  $c^i$  of the known  $C$  classes. To obtain the novelty score of a test sample  $\mathbf{x}^*$ , we project  $\mathbf{x}^*$  into null subspace and hence becomes the target point  $t^*$ . We use the smallest Euclidean distance between  $t^*$ , and the other target points from the training set:  $\mathbf{t}^1, \mathbf{t}^2, \dots, \mathbf{t}^C$  to obtain the novelty score of  $\mathbf{x}^*$ :

$$\text{Min}_{1 \leq i \leq C} \text{dist}(t^*, t^i) \quad (2.1)$$

(b) In one class novelty detection (figure 2.6 - b), we test if a sample belongs to a class  $c^1$  or not. Its not possible to use multi-class novelty detection as it leads to zero projection direction in space. We use one-class methods similar to one-class SVM [16]. All class samples will be then mapped into a single target value  $t$  along a single null direction. We again set a certain threshold to take the novelty decision:

$$|t - t^*| \quad (2.2)$$

This approach checks if a sample belongs to a known category or class from the training set or to a new, unseen class. We can refer to as a novelty detection method. Similar to abstraction based novelty detection, it also transforms a novelty detection problem into an outlier detection problem by projecting the data into a different hyper-dimensional space, the null-kernel space. However, this approach differs from our own term definition of novelty detection represented in 2.3. This approach focuses on a kernel-spaced method and is expected to follow our term definition that focuses on abstraction based methods in particular.

## An example of abstraction-based method

”Outside the box” is a white-box abstraction-based novelty detection method proposed in [6]. Abstraction-based novelty detection involves 2 phases: (1) an

---

**Algorithm 1: Constructing abstraction at layer  $\ell$**

---

**Input:**  $\mathcal{Y}$ : output classes  
 $D = \{(\vec{x}^{(1)}, y^{(1)}), \dots, (\vec{x}^{(m)}, y^{(m)})\}$ : training data  
 $\tau$ : clustering parameter

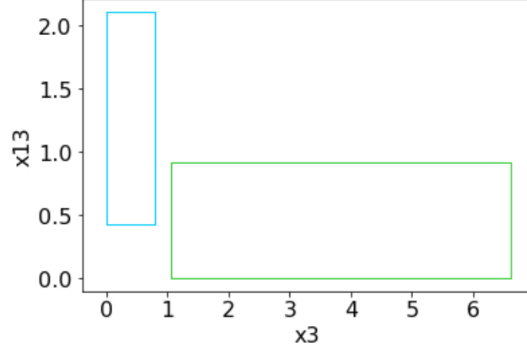
**Output:**  $A_1, \dots, A_{|\mathcal{Y}|}$ : lists of abstractions

```

1 for  $y \in \mathcal{Y}$  do
  // collect all outputs at layer  $\ell$  for inputs of class  $y$ 
2  $W_y \leftarrow \left\{ \text{watch}(\vec{x}, \ell) \mid (\vec{x}, y) \in D \wedge y = \text{classify}(\vec{x}) \right\}$ ;
3  $C_y \leftarrow \text{cluster}(W_y, \tau)$ ; // divide collected vectors into clusters
4  $A_y \leftarrow []$ ; // list of abstractions for class  $y$ 
5 for  $C \in C_y$  do
  // construct abstraction for vectors in cluster  $C$ 
6  $A_y^C \leftarrow \text{abstract}(C)$ ;
7  $A_y.\text{add}(A_y^C)$ ; // add abstraction to list
8 end
9 end
10 return  $A_1, \dots, A_{|\mathcal{Y}|}$ 

```

---



(a) Abstraction Phase

---

**Algorithm 2: Monitoring at layer  $\ell$**

---

**Input:**  $\vec{x}$ : network input  
 $A_1, \dots, A_{|\mathcal{Y}|}$ : lists of abstractions

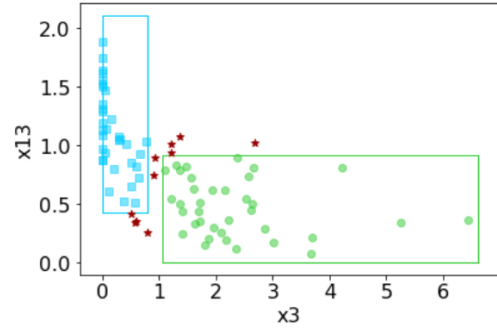
**Output:** "accept"/"reject": answer

```

1  $y \leftarrow \text{classify}(\vec{x})$ ; // predict class of  $\vec{x}$ 
2  $\vec{v} \leftarrow \text{watch}(\vec{x}, \ell)$ ; // collect output at layer  $\ell$ 
3 for  $A_y^C \in A_y$  do // check each abstraction for class  $y$ 
4   if  $\vec{v} \in A_y^C$  then
5     return "accept"; // found an abstraction containing  $\vec{v}$ 
6   end
7 end
8 return "reject"; //  $\vec{v}$  is not contained in any abstraction

```

---



(b) Monitoring phase

Figure 2.7: An example of novelty detection system with an abstraction based approach

abstraction phase and (2) a monitoring phase. Constructing an abstraction at layer  $l$  of the monitored network for class  $y$  works as follows (abstraction phase):

1. Collect outputs at layer  $l$  for inputs of class  $y$ .
2. Divide collected vectors into clusters.
3. Construct an abstraction for vectors in each cluster, e.g. an enclosing box.

A monitor then takes a one-by-one decision on each testing sample and identifies

it as either valid (i.e. the classifier prediction is correct on assigning the sample to one the training classes), or invalid (i.e. the classifier prediction is rejected). The monitor decision is based on verifying whether a special representation of the sample falls within one of the boxes constructed during training or outside these boxes. The monitoring stage at layer  $l$  hence works as follows (monitoring phase):

1. Predict class of input  $\mathbf{x}$ .
2. Collect output at layer  $l$  into a vector  $\mathbf{v}$ .
3. Check if any abstraction belonging to the predicted class contains  $\mathbf{v}$ .
4. The prediction is invalid if the check returns empty, otherwise valid.

We represent the 2 phases of the algorithm in figure [2.7](#). This approach uses several benchmarks for testing. It also tunes different environments. At the monitoring stage, it can include multiple layers rather than a single layer, by constructing additional monitors; one for each layer, or concatenating all outputs to pass into a single monitor. It's also possible to shrink or enlarge boxes according to a factor value  $\alpha$  tolerance. This algorithm also considers other types of abstractions, for example, balls and octagons.

## 2.4 Security risk

Results of all proposed approaches were proved to be very effective in terms of accuracy or reliability. Despite this, it can be critical when used in the real world setting. We can't guarantee that these techniques are robust to attacks



and always free of malicious attempts.

Little or none work has been conducted on the security of the aforementioned approaches either in the presence of adversarial samples for fooling the classifier or especially against crafted samples for fooling the novelty detector and the classifier at the same time. We consider security an important factor, for our study, we show the security use-case of "outside the box" [6] technique and analyse it in different ways.

We take abstraction based novelty detection ("outside the box") as a case-study in this thesis and show that: (1) its monitors are not efficient when adversarial testing samples are presented, and (2) these monitors can themselves be attacked by appending the adversarial generation process with new constraints. In other words, these boxes could be penetrated. Moreover, we adapt a suitable defense mechanism that seeks to minimize the risk posed by adversarial examples by training on both adversarial and clean data.

# Chapter 3

## Methodology

It was shown that "outside the box" is an efficient novelty detection approach used in many applications. However, we have already mentioned its weak security. This approach can be exposed to adversarial attacks that lead the monitor to output wrong decisions. We distinguish two types of attacks against this schema:

**Attack 1 - from valid to invalid** Consider a testing input  $\mathbf{x}$  which would normally be identified as valid by the monitor and belongs to one of the training classes. This attack modifies  $\mathbf{x}$  in a slight and unperceivable way so that it gets rejected by the monitor. An example of this attack, would be to imagine a denial of service for a legitimate user in a face recognition system.

**Attack 2 - from invalid to valid** Consider a testing input  $\mathbf{x}$  which would normally be rejected by the monitor as not belonging to any of the classes seen during training. This attack modifies  $\mathbf{x}$  in a slight and unperceivable way to make it get accepted by the monitor. The attack can be targeting a preset prediction, or just going with any prediction output by the neural network.

As an example of such attack, we would imagine letting go an intruder in a face recognition system. The intruder is identified as any of the legitimate white-list users.

In terms of implementation, we propose to formulate each attack as an optimisation problem that can be solved by an iterative optimisation algorithm. We also experiment with off-the-shelf adversarial attacks against neural networks and assess their efficiency, as well as augmenting them by an optimisation component to target a specific attack and make them more efficient. We detail these two approaches next.

### 3.1 Optimisation based attacks

#### Attack 1: from valid to invalid

Consider a neural network that is trained over only two classes, where each class is represented by the monitor under one box. As shown in figure 3.1 - a, we push the images of valid points, as represented by the monitor, from both classes  $\blacksquare$  and  $\bullet$  to fall outside their boxes and therefore be marked as novel exactly as for the  $\star$  points. Note that points are still marked as their class symbols ( $\blacksquare$  or  $\bullet$ ) after the attack just for a simpler representation. All samples that are located outside their class abstraction boxes are rejected by the monitor and marked as  $(\star)$ .

More generally, consider a point  $\mathbf{x}^0$  such as  $\text{monitor}(\mathbf{x}^0, c) = 1$  (accept as class  $c$ ), our goal is to find a point  $\mathbf{x}$  as close as possible to  $\mathbf{x}^0$  such that  $\text{monitor}(\mathbf{x}, c) = 0$  (reject). For measuring the distance between the two points we either use the

$L_1$  or  $L_2$  norm difference. In addition, we require that  $\mathbf{x}$  preserves the same prediction as of  $\mathbf{x}^0$  via the monitored network:  $\text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0) = c$ .

In case of the  $L_1$  norm, we replace the non-differentiable objective function by a differentiable one through introducing a new vector  $\mathbf{z}$  having the same dimension as  $\mathbf{x}$ . We now need to minimize the sum of  $\mathbf{z}$  vector instead of minimizing  $L_1$  norm objective. Our non-differential problem hence becomes a linear differential problem as follows:

$$\begin{aligned}
& \text{Minimize} && \sum_{i=1}^n z_i \\
& \text{Subject to} && z_i + (x_i - x_i^0) \geq 0 \\
& && z_i - (x_i - x_i^0) \geq 0 \\
& && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 0 \\
& && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
\end{aligned} \tag{3.1}$$

We next use the  $L_2$  norm ( $\|\mathbf{x} - \mathbf{x}^0\|_2$ ) to formulate a second optimisation problem. The constraints for  $L_2$  are same as for  $L_1$  except that  $L_2$  norm is already a differential objective function and can be directly tuned by a linear solver. The  $L_2$  norm attack focuses on minimizing the square root of the sum of squared differences of  $(\mathbf{x}^0)$  and  $(\mathbf{x})$  elements. We then get a linear programming problem as follows:

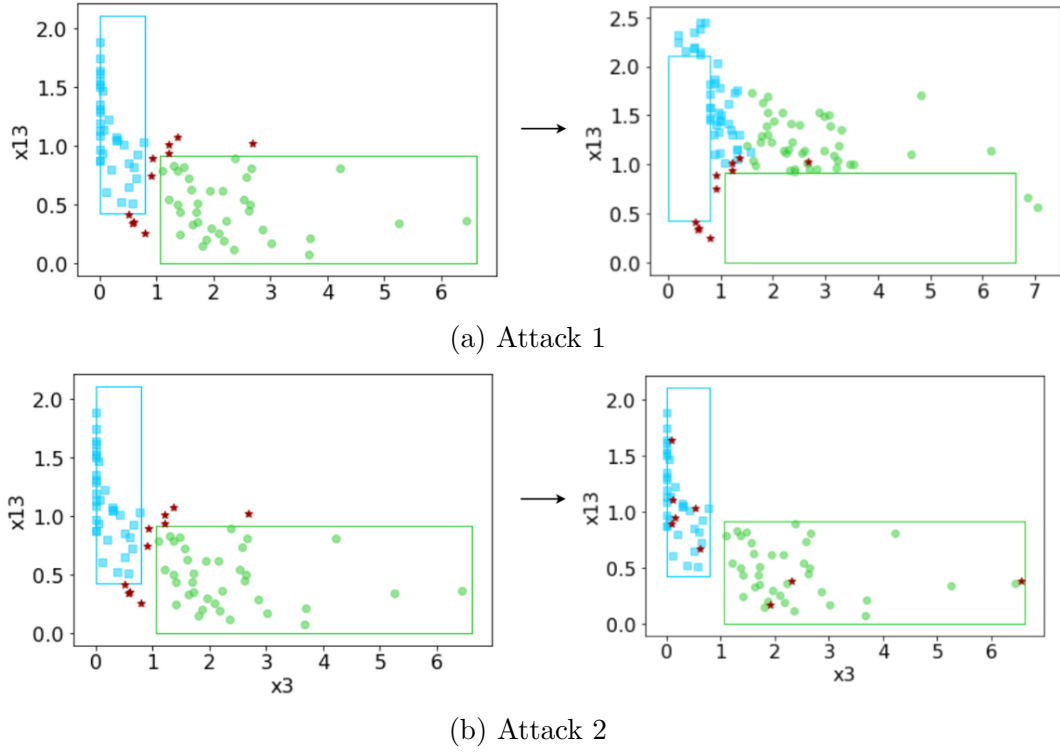


Figure 3.1: Optimisation attack generalization

$$\begin{aligned}
 & \text{Minimize} && \sqrt{\sum_{i=1}^n |x_i - x_i^0|^2} \\
 & \text{Subject to} && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 0 \\
 & && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
 \end{aligned} \tag{3.2}$$

## Attack 2: from invalid to valid

In this attack, we push the images of invalid points  $\star$ , as represented by the monitor, towards the boxes representing legitimate classes. The points will be marked by the monitor as valid if, at the same time, the neural network prediction matches the box owner, either as  $\blacksquare$  or  $\bullet$ . The 2d projection of the monitor representation for a binary classifier is shown in figure [3.1](#) - b. Note again that

points are still marked as invalid ( $\star$ ) after the attack just for an easier representation. Any point that belongs to a box of its same class is accepted by the monitor and marked as its class symbol ( $\blacksquare$  or  $\bullet$ ).

Now given a point  $\mathbf{x}^0$  where  $\text{monitor}(\mathbf{x}^0, c) = 0$  (reject), our goal is to find a point  $\mathbf{x}$  as close as possible to  $\mathbf{x}^0$  such that  $\text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1$  (accept as same class of  $\mathbf{x}^0$ ). For measuring the distance between the two points we also choose the  $L_1$  norm or the  $L_2$  norm difference.

For the  $L_1$  norm, we formulate the following problem:

$$\begin{aligned}
& \text{Minimize} && \sum_{i=1}^n z_i \\
& \text{Subject to} && z_i + (x_i - x_i^0) \geq 0 \\
& && z_i - (x_i - x_i^0) \geq 0 \\
& && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1 \\
& && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
\end{aligned} \tag{3.3}$$

For the  $L_2$  norm, we formulate the following problem:

$$\begin{aligned}
& \text{Minimize} && \sqrt{\sum_{i=1}^n |x_i - x_i^0|^2} \\
& \text{Subject to} && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1 \\
& && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
\end{aligned} \tag{3.4}$$

The four formulated problems can be efficiently solved by constrained optimisation numerical methods that are either local such as SLSQP [17] and COBYLA [18] or global such as Differential Evolution (DE) [19] and SHGO [20].

- Local optimisation: Search over a specific region
  - SLSQP: Is a local optimisation method that focuses on minimizing a scalar function of one or more variables using Sequential Least Squares Programming (SLSQP) . SLSQP is gradient based (white-box) and accepts both equality and inequality constraints. It finds a feasible point by applying gradient steps along a positive direction. It performs a local search. It requires an initial starting point and assumes the solution is close in the local region.
  - COBYLA: Is a local optimisation method that focuses on minimizing a scalar function of one or more variables using the Constrained Optimisation BY Linear Approximation (COBYLA) algorithm . Unlike SLSQP, it is gradient free (black-box) and only accepts inequality constraints. It finds a feasible point by evaluating the objective function and constraints via a trust region. It also requires an initial guess and considers the solution is present within a desired region.
- Global optimisation: Search over wider parameter space
  - SHGO: Is a global optimisation method that focuses on finding the global minimum of a function using SHGO optimisation (Simplicial Homology Global Optimisation). It is gradient free (black-box) and accepts both inequality and equality constraints. It uses methods that detect the homological properties of the objective function. It is mainly based on pure Combinatorial Theory. It is appropriate for computationally expensive black box functions. That is, for every input the

output is evaluated and solution is given respectively.

- Differential Evolution (DE): Is a global optimisation method that focuses on finding the global minimum of a multivariate function, function who has multiple input outputs. It is gradient free (black-box) but often requires many function evaluations than gradient based methods. It is a genetic population based method. It works via mutating inputs with other input solutions to create other trial inputs.

We used implementations from SciPy library [21] to solve our formulated problem using methods proposed. We show-case the whole experiment of attacks as detailed in chapter 5.

## 3.2 Adversarial attacks against neural networks

Another idea is to use known adversarial attacks against neural networks as a starting point for attacking the monitor. Adversarial neural network attacks aim at changing the prediction of an input  $\mathbf{x}^0$  when replaced by a close point  $\mathbf{x}$  as of  $\text{predict}(\mathbf{x}) \neq \text{predict}(\mathbf{x}^0)$ . In their white-box primitive version, a step is taken in the opposite direction of the gradient of the objective function at the point  $\mathbf{x}^0$ . We use the implementation of both white-box and black-box adversarial attacks from the Foolbox library [22]. We use different adversarial attacks and represent the description of each using the table 3.1.

It is interesting to check whether adversarial samples would be detected as novel by the novelty monitor. In case these samples are not, we may consolidate by optimisation based methods to make them pass as valid points. In other words,



we can take an adversarial sample as a starting point for our search for an attack against both the neural network and the monitor.

Attack	Description
FGSM	This attack computes the gradient of loss with order $L_\infty$ . Gradient steps take the direction that maximizes the loss of image.
$L_2$ PGD	This attack computes the gradient and takes the direction of greatest loss, except it starts at a random perturbation in the $L_2$ ball.
$L_2$ BasicIterativeAttack	This attack applies the same FGSM algorithm multiple times with smaller step size.
$L_2$ FastGradientAttack	This attack extends the FGSM to $L_2$ norm.
InversionAttack	This attack inverts negative images by inverting the pixel values.
BinarySearchContrastReductionAttack	This attack reduces the contrast of the input using a binary search to find the smallest adversarial perturbation.
$L_\infty$ DeepFoolAttack	This attack takes gradient steps in the direction of the image class having the smallest distance of $L_\infty$ norm.
SaltAndPepperNoiseAttack	This attack adds white or black pixels until the input is mis-classified.
$L_0$ BrendelBethgeAttack	This attack follows the adversarial boundary to find the minimum distance $L_0$ to the clean image.

Table 3.1: Foolbox neural network adversarial attacks and description

Our goal here is to find a point  $\mathbf{x}$  very close to  $\mathbf{x}^0$  such that  $\text{predict}(\mathbf{x}) \neq \text{predict}(\mathbf{x}^0)$  and  $\text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1$  (accept). We assess the performance of an abstraction based monitor to flag adversarial samples as novel. We study the effect of tuning "outside the box" parameters to enhance the robustness of the monitor. In a second time, we use optimisation techniques to force adversarial samples to bypass the monitor.

### 3.3 Defense mechanism against attacks

Defending adversarial attacks is a critical step towards reliable deep learning models. We aim to propose existing defense mechanisms that require appending noisy data to an auto-encoder so that it reconstructs original samples. Experimental results show that this approach could be successful at the first step and could increase the robustness of the monitor against adversarial attacks. We provide an analysis on a defense mechanism known as denoising auto-encoders.

Auto-encoders are some type of neural networks just like classifiers. Classifiers condense all data of a single image into a single label, auto-encoders however compress the data into a latent vector, known as  $z$ , as stated earlier the latent space. Auto-encoders main goal is to preserve the opportunity to re-create same exact images in the future. While re-creating again the same data may seem an impossible task, auto-encoders can actually do this task. An auto-encoder consists of two networks: an encoder and decoder, and a middle layer mapping inputs into a lower dimension, called a bottle-neck layer. The role of encoders is to compress a representation, for example an image, into a latent vector  $z$ , then a decoder reconstructs back an image from a latent vector  $z$ , i.e., exactly the

opposite.

We realize that optimisation or neural network adversarial attacks usually end up producing noisy images. Noise can sometimes be catastrophic and it can lead to a change in the validity or mis-classification of inputs. So we tend to use denoising auto-encoders in specific. Denoising auto-encoders help to reconstruct the clean version of an image from corrupted and noisy images (see figure [3.2](#)). By feeding the encoder noisy images and setting the original clean images as our desired outputs, an auto-encoder learns to remove noise from these images. Images then become clear and succeed to preserve the original class and validity of inputs, hence hindering the effect of attacks.

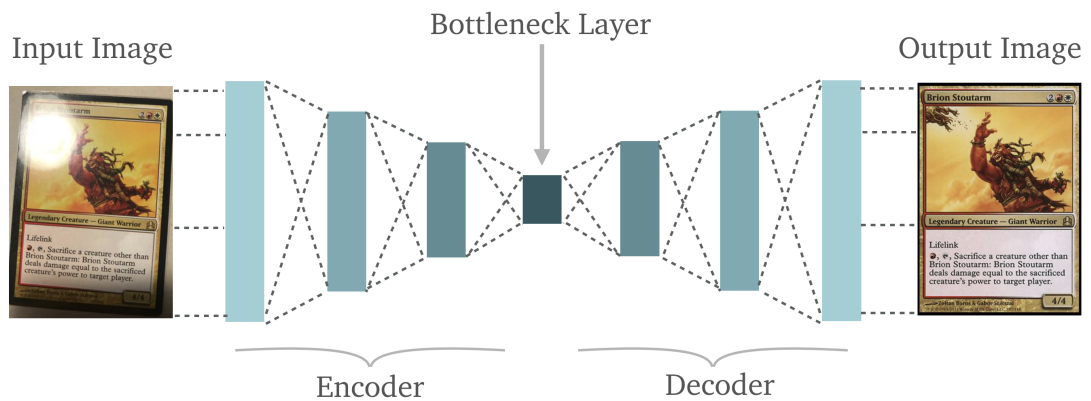


Figure 3.2: Denoising auto-encoders architecture

# Chapter 4

## Experimental setup

### 4.1 Attacks experimental setup

In optimisation solvers experiment, we test our proposed attacks with different network architectures, namely two very simple neural networks and a MNIST classifier. We construct network 1 as a modified version of a XOR network with 2 hidden layers A and B. We choose that network 1 is a two-class neural network, that involves only 2 classes: classes 0 and 1. Figure [4.1](#) represents its network architecture. We follow same abstraction based novelty detection procedure on network 1 to build a monitor and construct the abstraction boxes using the training inputs. Later during testing, inputs are accepted if they belong to a box having same output class otherwise rejected accordingly.

We prepare decision plots instead of abstraction box plot paradigm to evaluate input samples. Decision boundary plots make it easier to notice the regions where inputs get accepted or rejected. We first construct 2 decision boundary plots that

distinguish the regions defining classification and validity of inputs, then combine these plots to construct an additional plot that limits the boundaries of both validity and classification. Figure 4.2 shows the decision boundary plot with the training inputs projection.

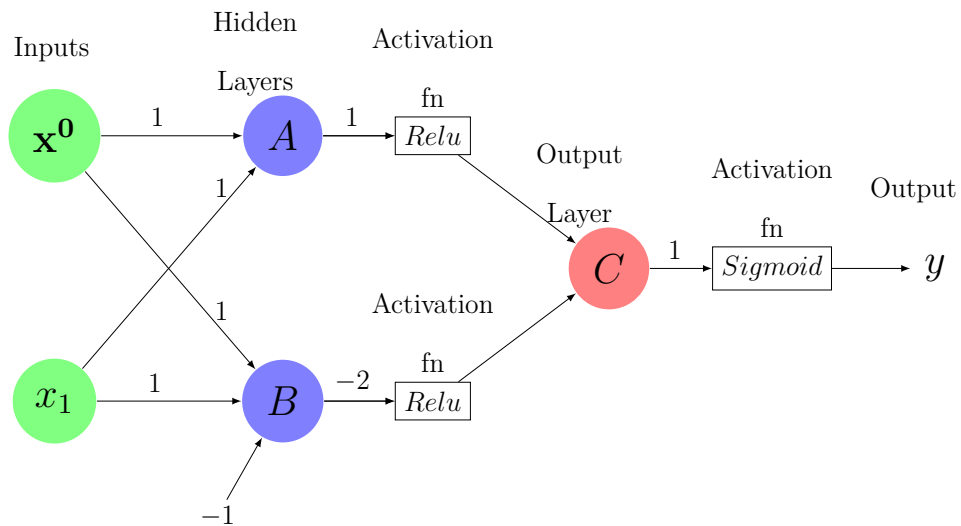
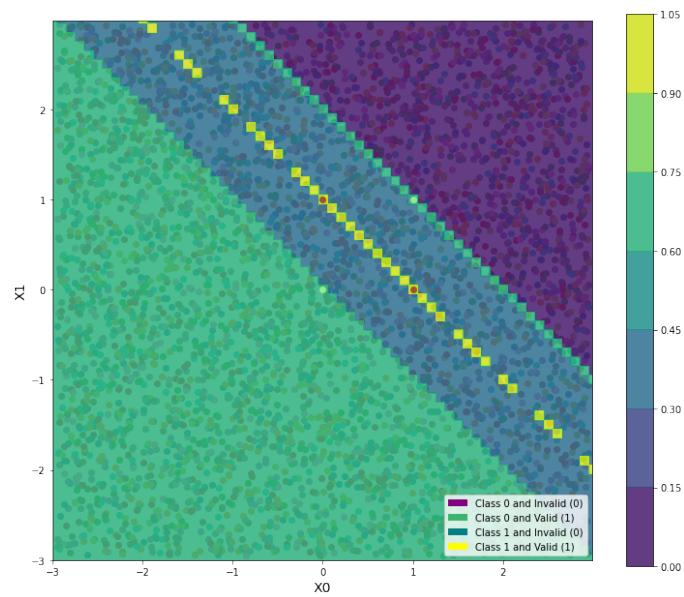


Figure 4.1: Network 1 architecture



(a) Classification & Validity Region

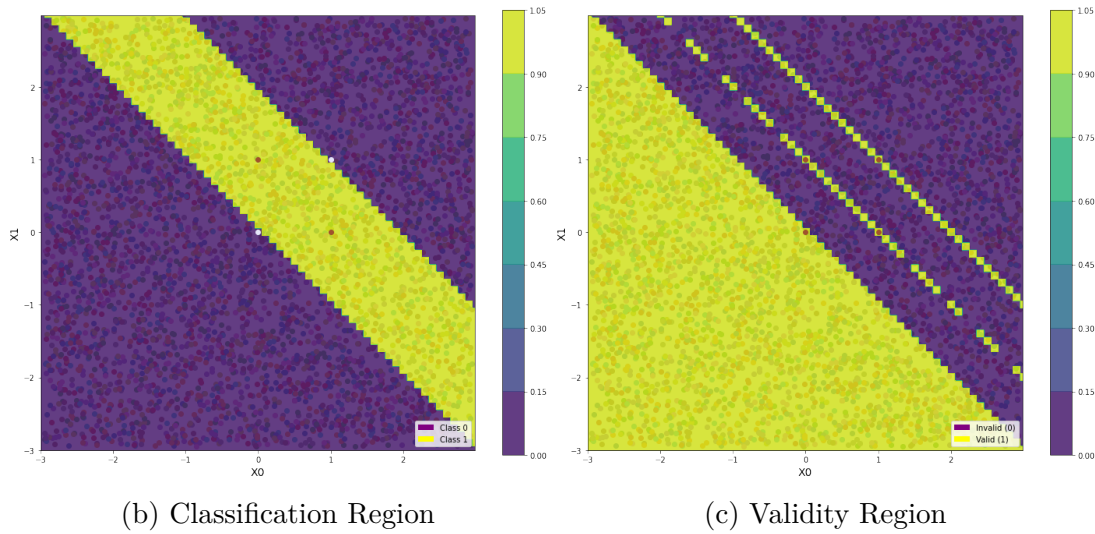


Figure 4.2: Decision Boundary plot for network 1

We consider another neural network, network 2, inspired from <https://playground.tensorflow.org/>. Network 2 has multiple layers, however, for simplicity, we consider outputs at layers A and B. Similar to network 1, network 2 is a two-class neural network that involves only 2 classes. The network architecture of network 2 is represented in figure 4.3. We follow same abstraction based novelty detection procedure on network 2 as for network 1.

We again construct the decision boundary plots that define the regions of classification and validity of inputs, in addition to projecting training inputs (figures 4.4). We also train a MNIST classifier once over 2 labels (0, 1) and once another on 3 labels (0, 1, 2) to evaluate for our experiment.

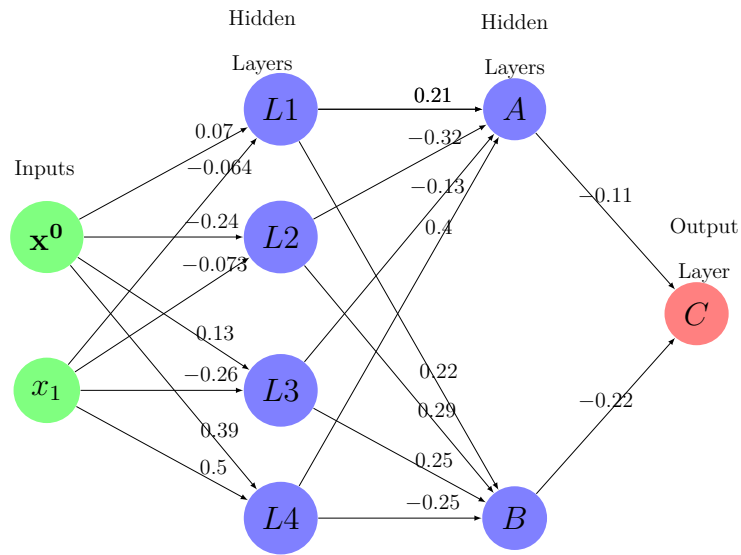
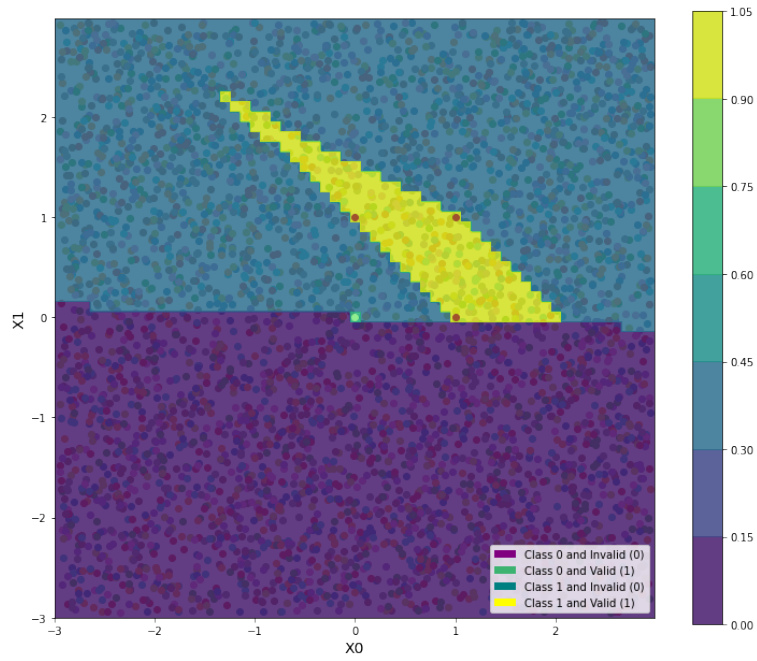


Figure 4.3: Network 2 architecture



(a) Classification & Validity Region



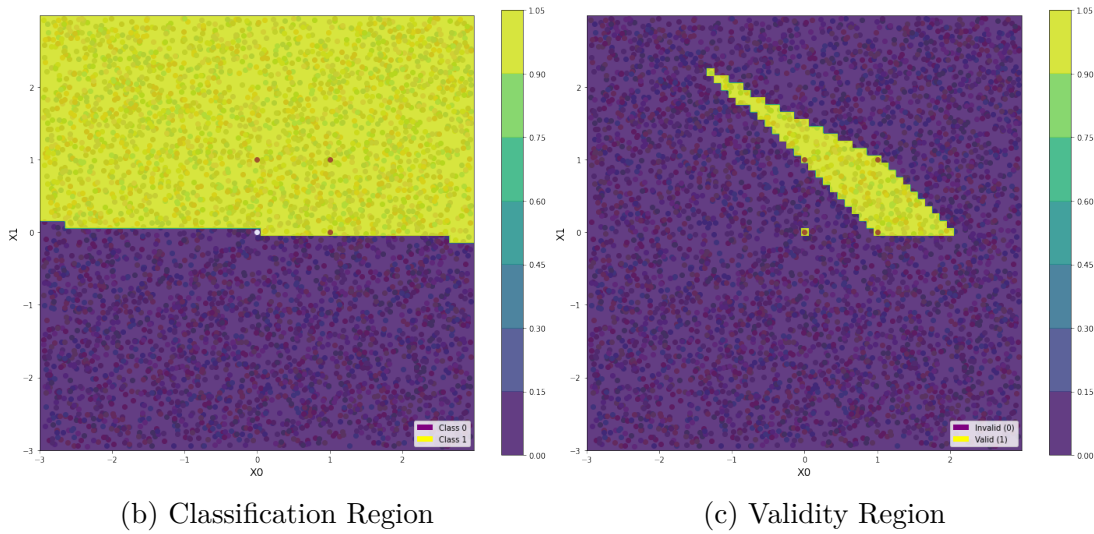


Figure 4.4: Decision Boundary plot for network 2

However, in adversarial attacks experiment, we only test on a MNIST classifier. We train the neural network of MNIST classifier over all the classes of the MNIST benchmark.

## 4.2 Defense experimental setup

For the second experiment, we need then to defend against the preceding attacks: optimisation and adversarial neural network based attacks. We first train a denoising auto-encoder on noisy images obtained by attacks. Once again, we consider a MNIST classifier trained over 2 labels and 3 labels, to protect against optimisation based attacks. We also consider a MNIST classifier trained over all the classes of the MNIST benchmark to test against adversarial neural network attacks.

# Chapter 5

## Optimisation solvers experiment

In this experiment, we evaluate the performance of different optimisation solvers in solving the four proposed optimisation problems and successfully generating adversarial samples. We evaluate four solvers from SciPy: COBYLA, SLSQP, SHGO and DE. We experiment with 4 network architectures described in chapter 4.

### 5.1 Attacks on network 1

#### 5.1.1 Attack 1 on network 1

We start by applying our attacks on network 1. As mentioned for attack 1, we need to pass a valid entity as invalid. We experiment attack 1 using local and global optimisation methods. We tested over 20 random  $\mathbf{x}^0$  samples while using  $L_1$  and  $L_2$  norm difference and counted the times where the optimiser found a solution, i.e. generated an effective and successful attack point  $\mathbf{x}$ . Results are

shown in figure 5.1.

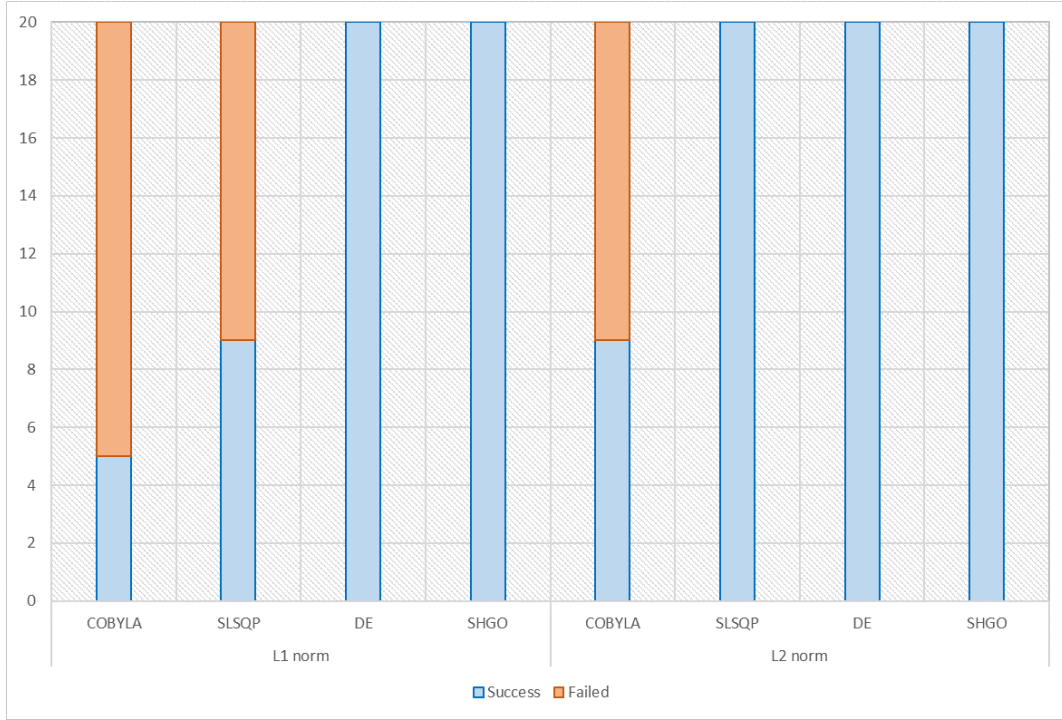
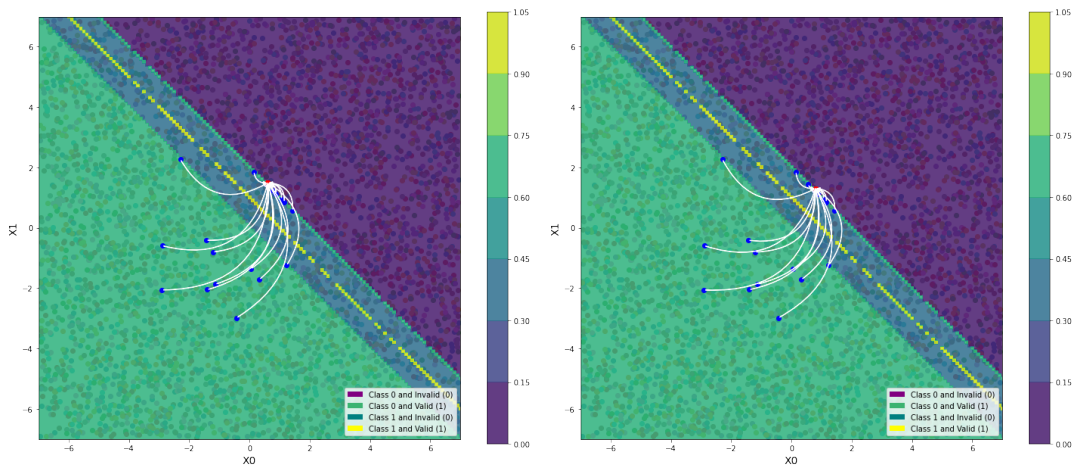
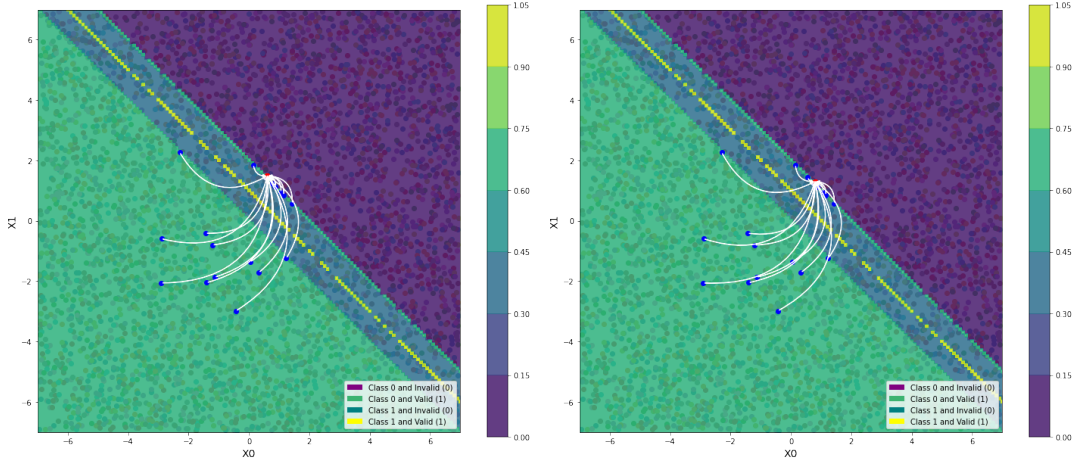


Figure 5.1: Success rate of different optimisation based methods for attack 1 on network 1



(a)  $L_1/L_2$  norm attack using SHGO



(b)  $L_1/L_2$  norm attack using DE

Figure 5.2: Decision boundary plot of attack 1 on network 1

Results show that DE and SHGO outperform other methods and succeed to convert all inputs from valid to invalid, using both norms. We plot attack results of DE and SHGO methods using the decision boundary plots established before in chapter 4. Figure 5.2 shows attack 1 where valid inputs (●) are pushed outside their boxes to be treated as invalid (●).

### 5.1.2 Attack 2 on network 1

We move now test attack 2, an invalid entity to act as valid. Similarly, we run attack 2 using both  $L_1$  and  $L_2$  norms. We choose 20 arbitrary inputs to convert into valid. We run the attack using the four optimisation methods. Results of attack 2 are represented in figure 5.3.

The overall result shows that DE is the most effective method. We project the attack attempts of DE method in figure 5.4. We see (●) invalid inputs transform to valid (●). (●) representation indicates the attack fails to find a solution.

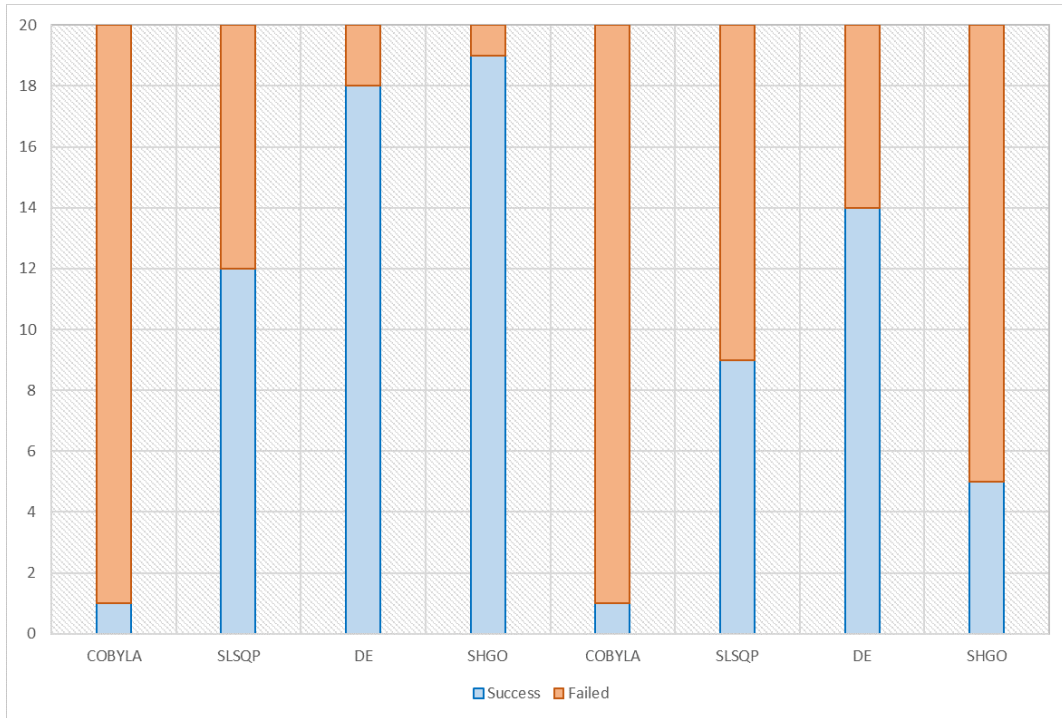
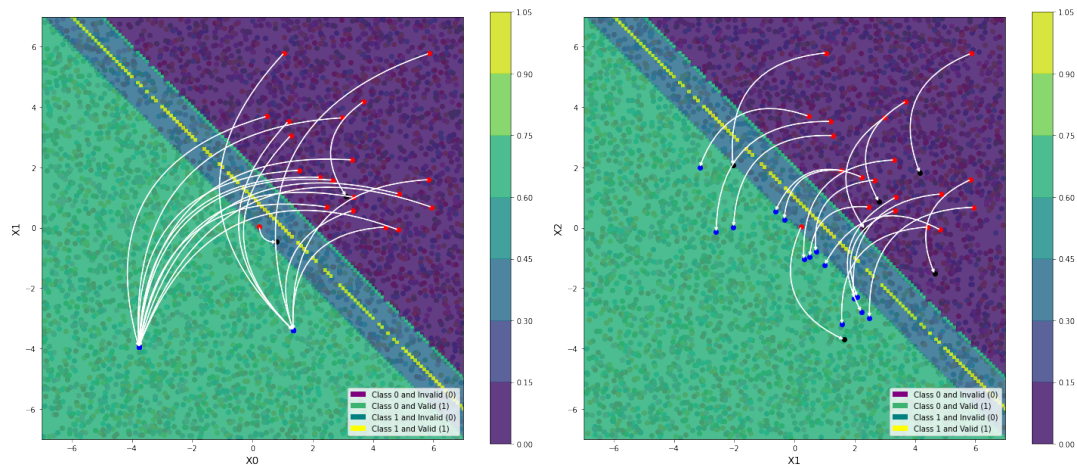


Figure 5.3: Success rate of different optimisation based methods for attack 2 on network 1



(a)  $L_1/L_2$  norm attack using DE

Figure 5.4: Decision boundary plot of attack 2 on network 1

## 5.2 Attacks on network 2

### 5.2.1 Attack 1 on network 2

We experiment the same attacks on network 2 as on network 1. We test attack 1 over 20 random  $\mathbf{x}^0$  samples and count the times where the optimiser found a solution, i.e. generated an effective attack point  $\mathbf{x}$ . We represent results in figure [5.5](#).

Results prove that all attacks are successful using all methods. However SLSQP, DE and SHGO perform better than COBYLA as an overall result of 20/20 successful samples, though there is a slight difference for COBYLA. We show the projection of attacks using SLSQP, SHGO and DE methods in figure [5.6](#).

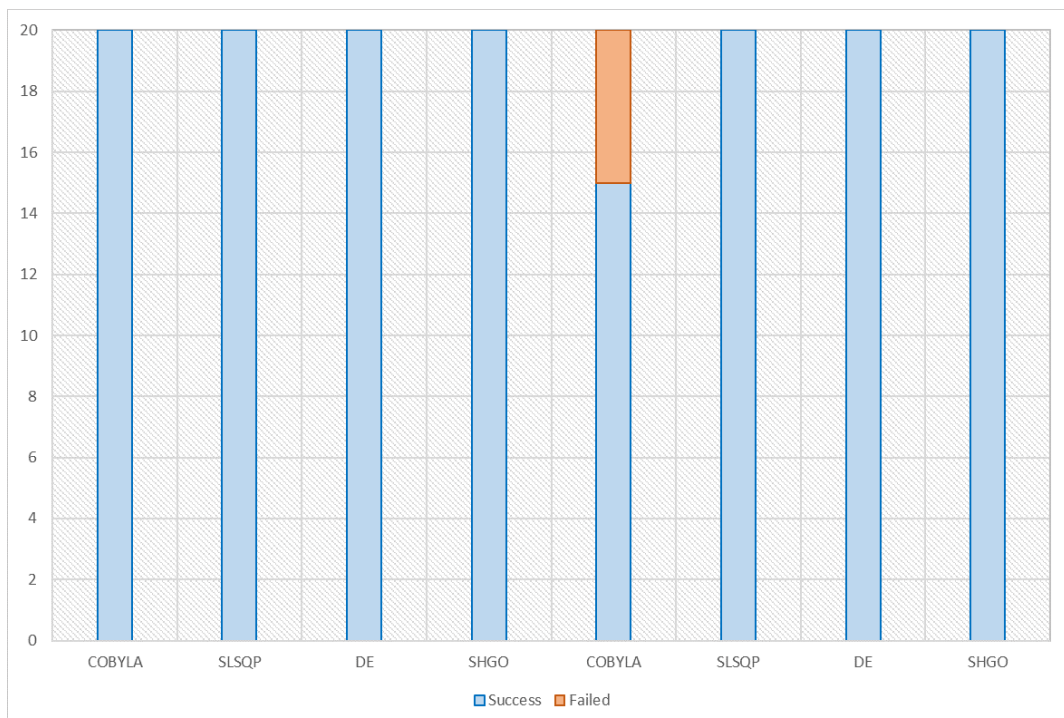
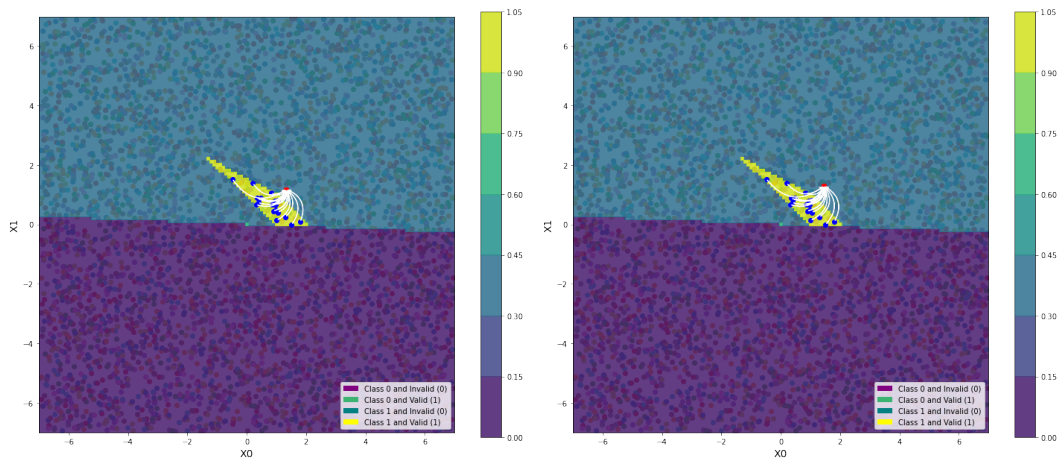
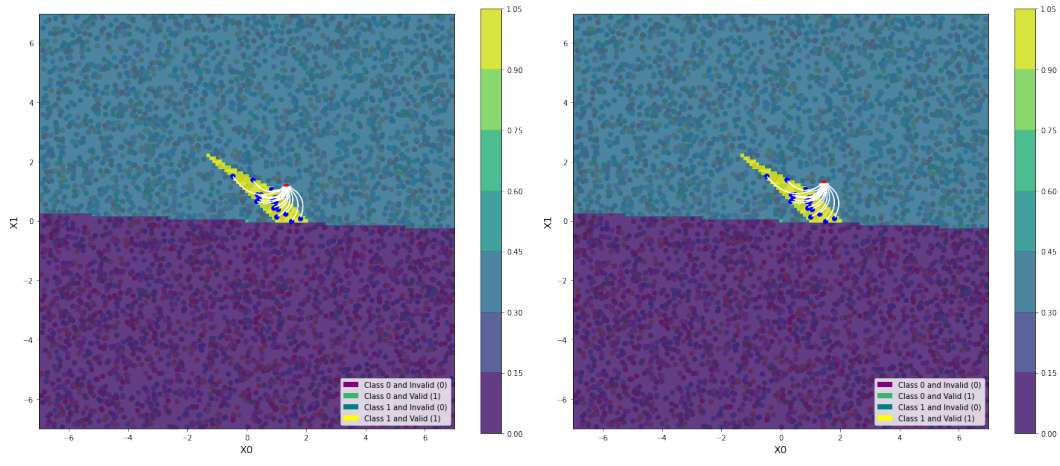


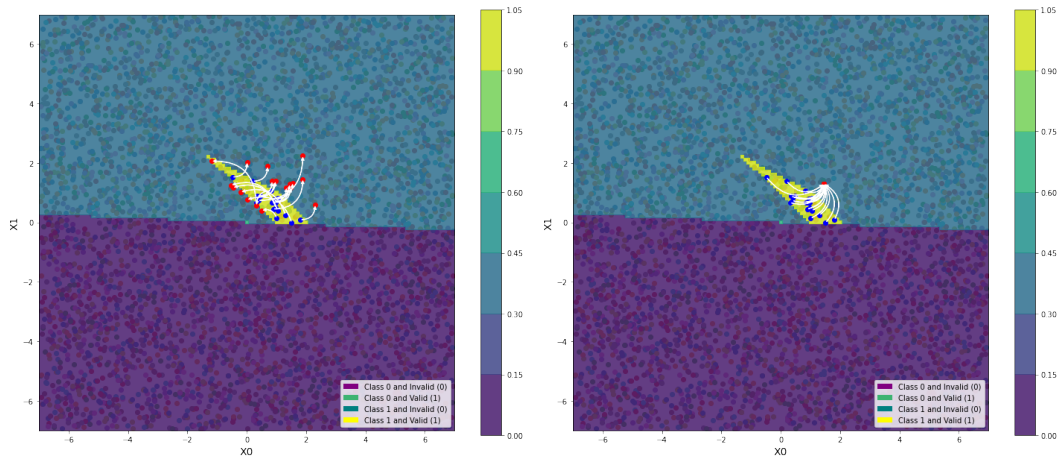
Figure 5.5: Success rate of different optimisation based methods for attack 1 on network 2



(a)  $L_1/L_2$  norm attack using SHGO



(b)  $L_1/L_2$  norm attack using DE



(c)  $L_1/L_2$  norm attack using SLSQP

Figure 5.6: Decision boundary plot of attack 1 on network 2

## 5.2.2 Attack 2 on network 2

We also test attack 2 over 20 random  $\mathbf{x}^0$  samples and represent results in figure [5.7](#). We notice that local optimisation methods like COBYLA didn't converge to a single solution. SLSQP also proved poor results using  $L_2$  norm. However global optimisation methods like SHGO and DE had better performance especially using  $L_1$  norm. We represent attack attempts of SHGO and DE methods using figure [5.8](#). The figures show all invalid points transform into valid.

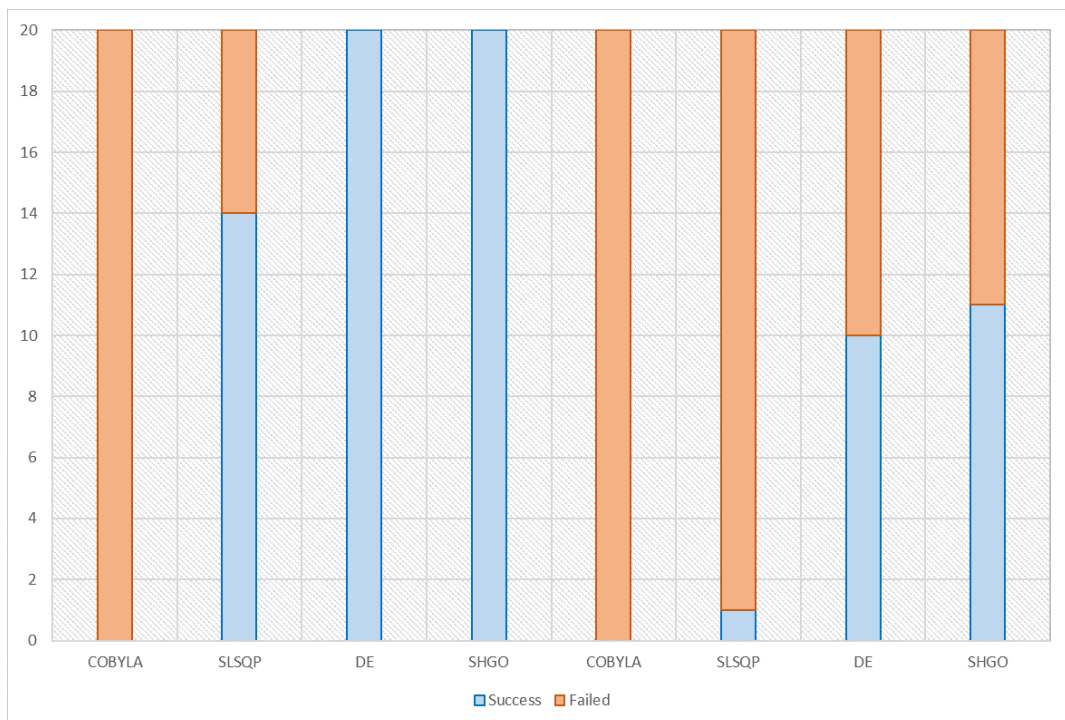
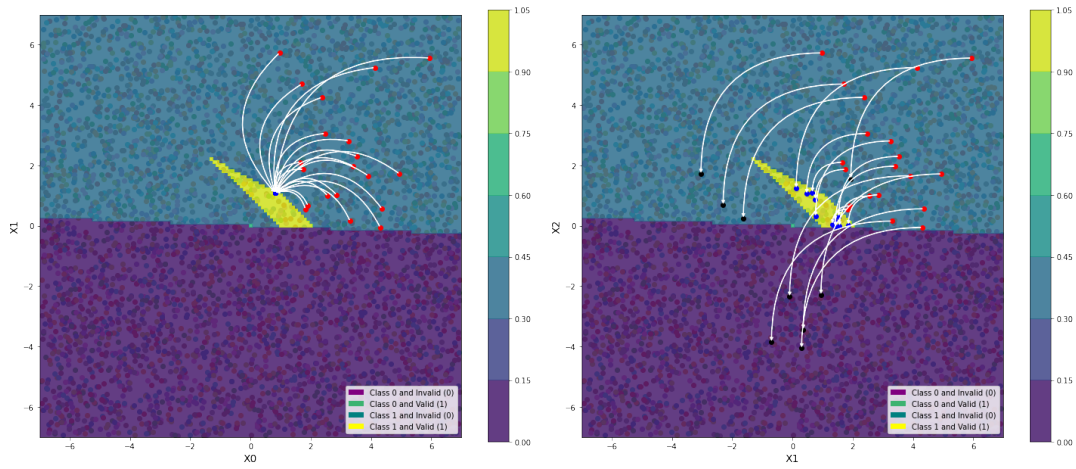
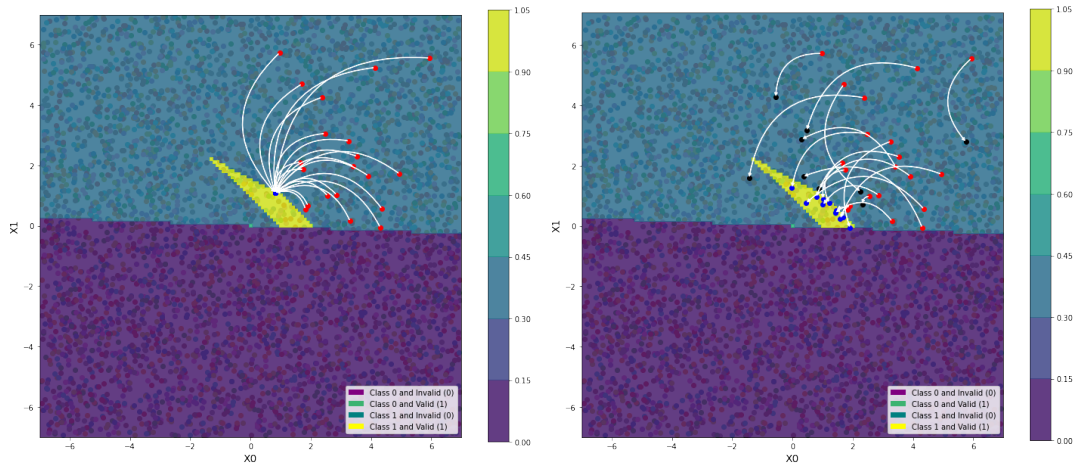


Figure 5.7: Success rate of different optimisation based methods for attack 2 on network 2





(a)  $L_1/L_2$  norm attack using SHGO



(b)  $L_1/L_2$  norm attack using DE

Figure 5.8: Decision boundary plot of attack 2 on network 2

## 5.3 Attacks on MNIST classifier

### 5.3.1 Attack 1 on MNIST classifier

We come to our main goal, to apply attacks on "outside the box" MNIST classifier to show its vulnerability to attacks. We follow the previous procedure of attacks but on a MNIST classifier. In attack 1, we find an invalid input  $\mathbf{x}$ , a perturbation

of a valid input  $\mathbf{x}^0$ , while preserving the class of  $\mathbf{x}^0$ . We test attack 1 using the four optimisation methods and using  $L_1$  and  $L_2$  norm attacks. We choose 20 arbitrary image samples for testing and represent successful attempts of attacks in figure 5.9 while training the neural network on both 2 and 3 classes.

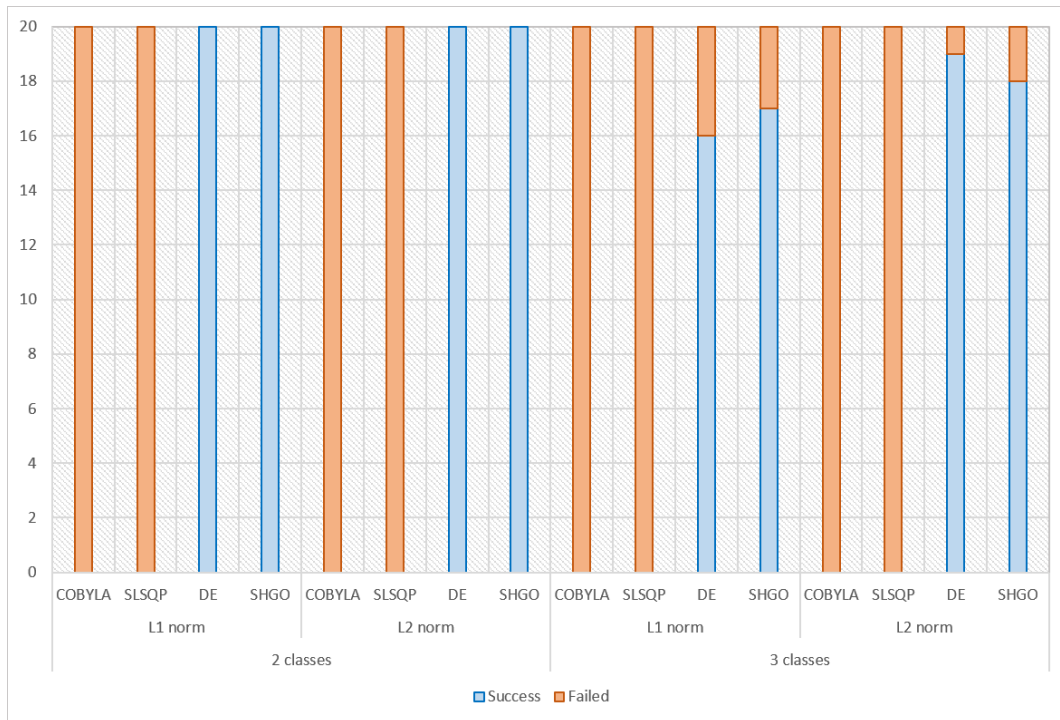
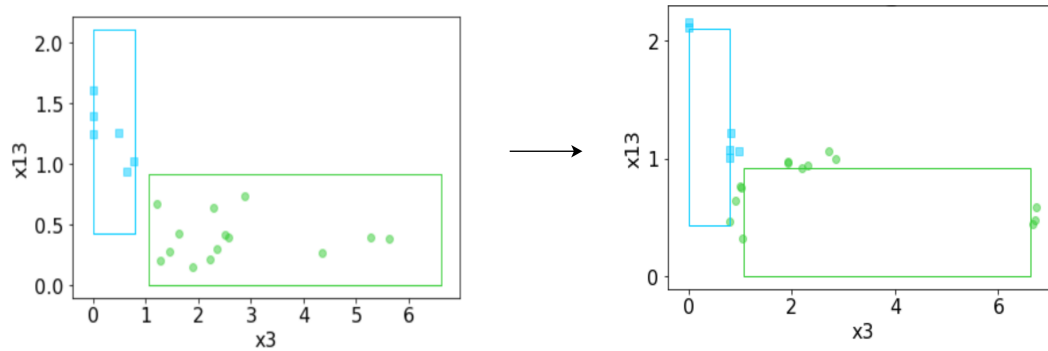
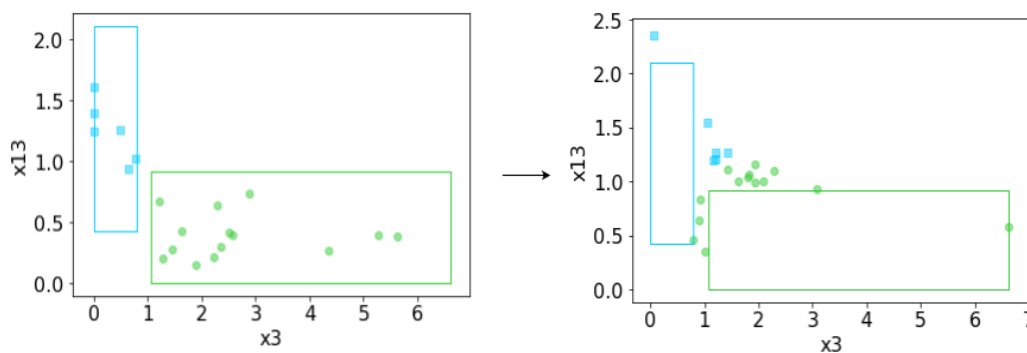


Figure 5.9: Success rate of different optimisation based methods for attack 1 on MNIST classifier

We see a deteriorating performance of COBYLA and SLSQP methods while testing on a MNIST classifier. These methods fail to converge to a feasible and compatible point. Global optimisation methods (DE and SHGO) however achieve very successful attempts. DE has better overall performance than SHGO when evaluating  $L_2$  norm results. However, when comparing DE and SHGO optimisers, we realize that images obtained by DE attack appear to be significantly noisy unlike SHGO which kept the original pattern of an image still recognizable.



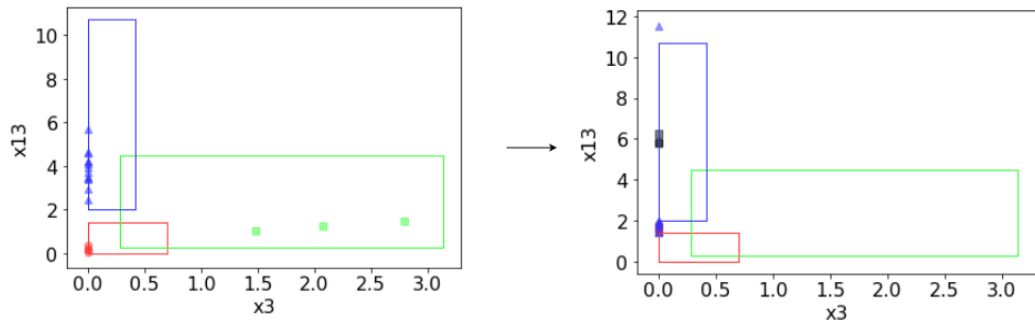
(a)  $L_1$  norm attack using SHGO



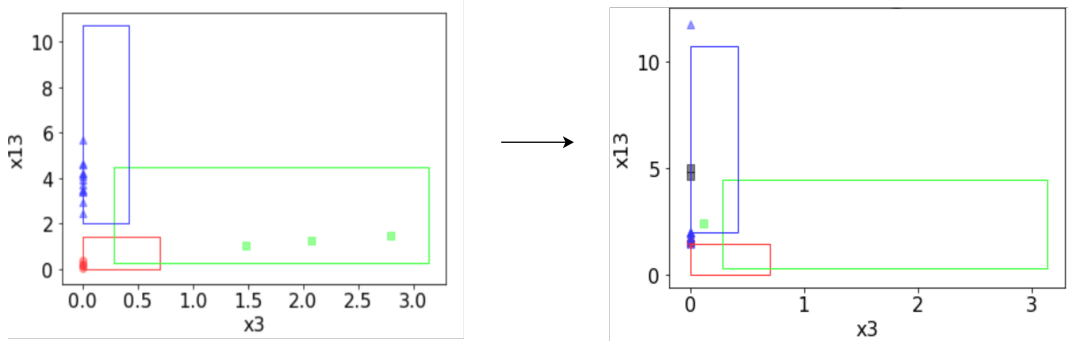
(b)  $L_2$  norm attack using SHGO

Figure 5.10: 2d projection of attack 1 attempts on MNIST classifier: 2 classes involved

We plot attack attempts using SHGO method in figure [5.10](#). The figure shows the transformation of inputs after attack 1 while the neural network is trained on 2 classes. We see the transformation of valid points (■ or ●) deviate outside their original box. The monitor now rejects these inputs and treats them as novel.



(a)  $L_1$  norm attack using SHGO



(b)  $L_2$  norm attack using SHGO

Figure 5.11: 2d projection of attack 1 attempts on MNIST classifier: 3 classes involved

We again show attack 1 via SHGO method while training the neural network on 3 classes (figure [5.11](#)). We see valid inputs (■, ▲ & ●) deviate outside their original box class. The monitor now rejects these inputs and treats them as novel. (★) represents that the attack fails to find a feasible solution. Despite that some of these (★) attempts deviate outside their box class, they fail to preserve their original class hence the attack represents a failure.

### 5.3.2 Attack 2 on MNIST classifier

We also apply attack 2 on MNIST classifier, we find a valid input  $\mathbf{x}$ , a perturbation of an invalid input  $\mathbf{x}^0$ , while preserving the class of  $\mathbf{x}^0$ . We experiment on 20 arbitrary invalid inputs and represent the success rate of these attempts in figure 5.12 while neural network is trained once on 2 classes and once on 3 classes.

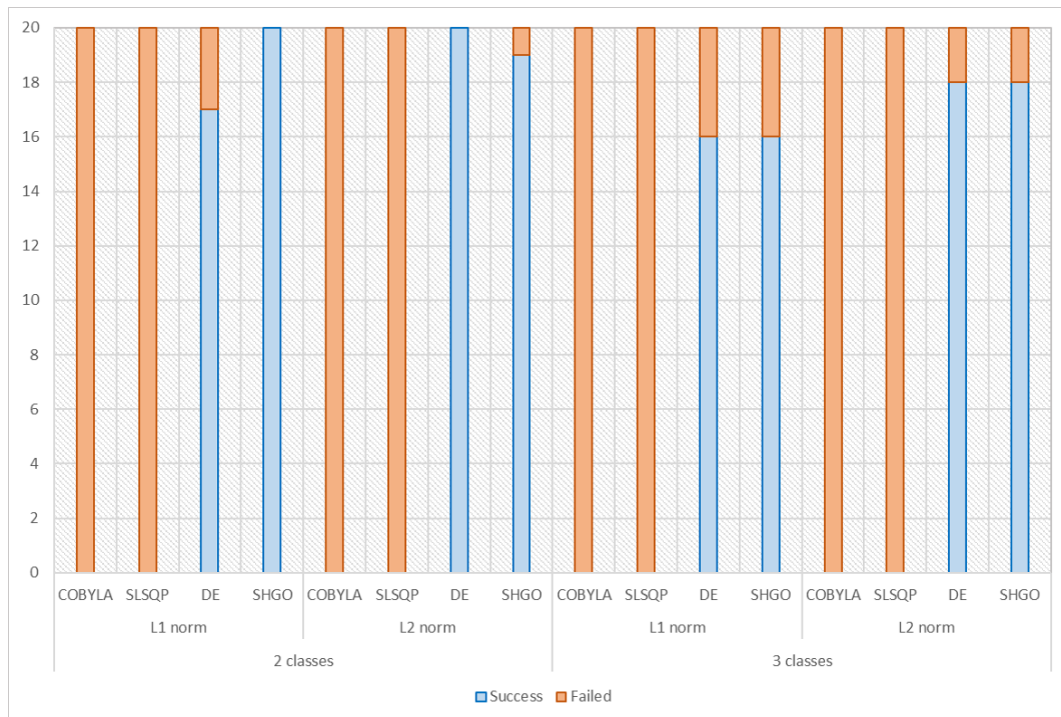
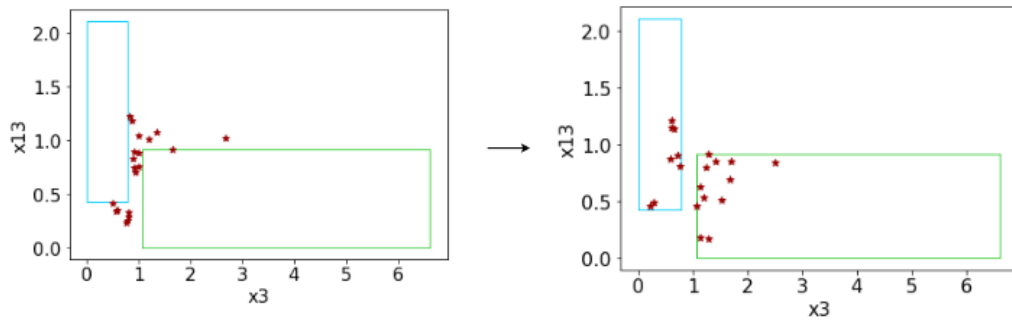
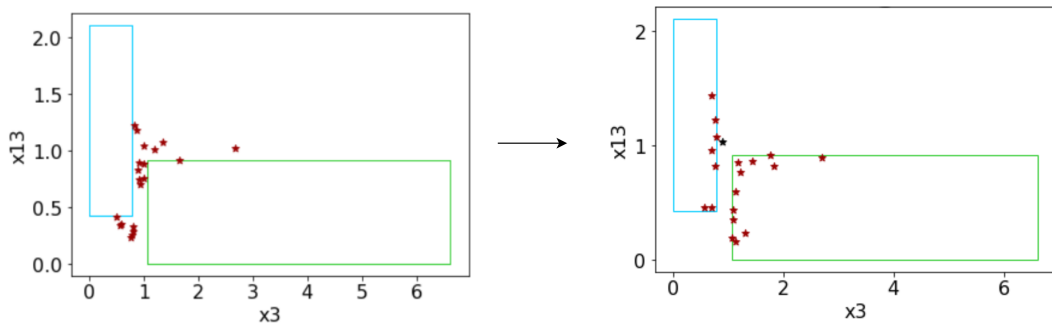


Figure 5.12: Success rate of different optimisation based methods for attack 2 on MNIST classifier

Still, local optimisation attacks (COBYLA and SLSQP) went unsuccessful while applying attack 2 on MNIST classifier. Global optimisation methods (DE and SHGO) prevail over other methods and prove good performance. DE and SHGO methods play a challenging role in attack 2. We stick to SHGO method as we consider again that image details are obscure using DE method, as we need the digit pattern to be still recognizable.



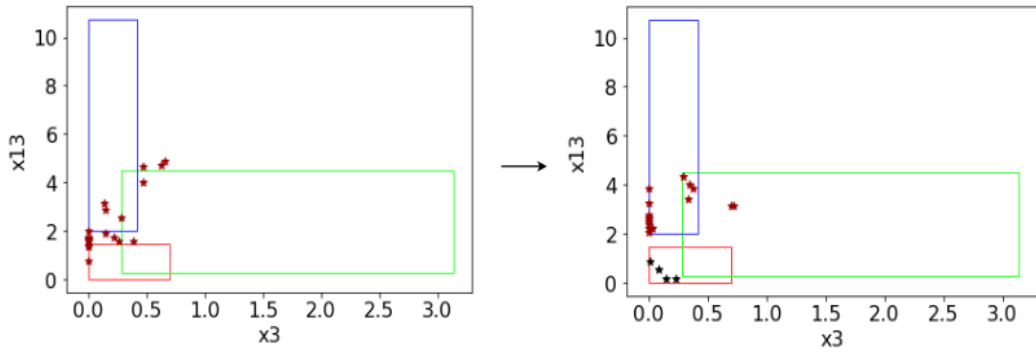
(a)  $L_1$  norm attack using SHGO



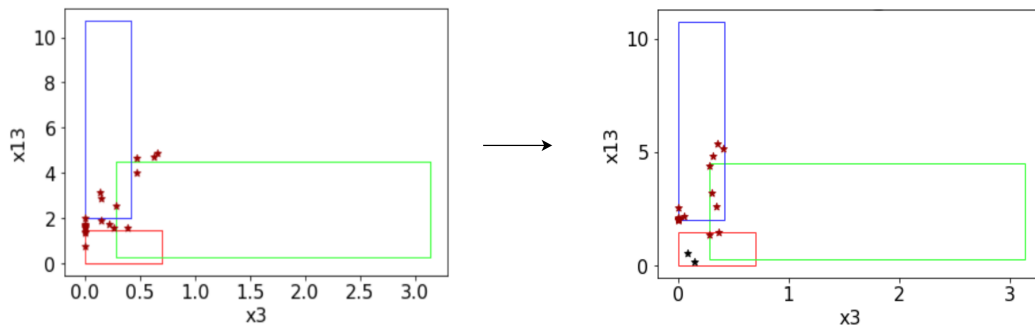
(b)  $L_2$  norm attack using SHGO

Figure 5.13: 2d projection of attack 2 attempts on MNIST classifier: 2 classes involved

We show the transformation of inputs using SHGO method while the neural network is trained on 2 classes in figure 5.13. We see that the invalid points ( $\star$ ) deviate inside the box of the same predicted class. Samples marked as ( $\star$ ) fail to belong to their box classes hence the monitor rejects these inputs.



(a)  $L_1$  norm attack using SHGO



(b)  $L_2$  norm attack using SHGO

Figure 5.14: 2d projection of attack 2 attempts on MNIST classifier: 3 classes involved

We show the results when neural network is trained on 3 classes. Figure 5.14 shows the attack attempts where inputs are transformed from invalid ( $\star$ ) to valid. Again, ( $\star$ ) represents that these points fail to exist inside their class boxes.

## 5.4 Discussion

In order to make it simpler for comparison, we combine the results of all mentioned attacks for all networks in figure 5.15.

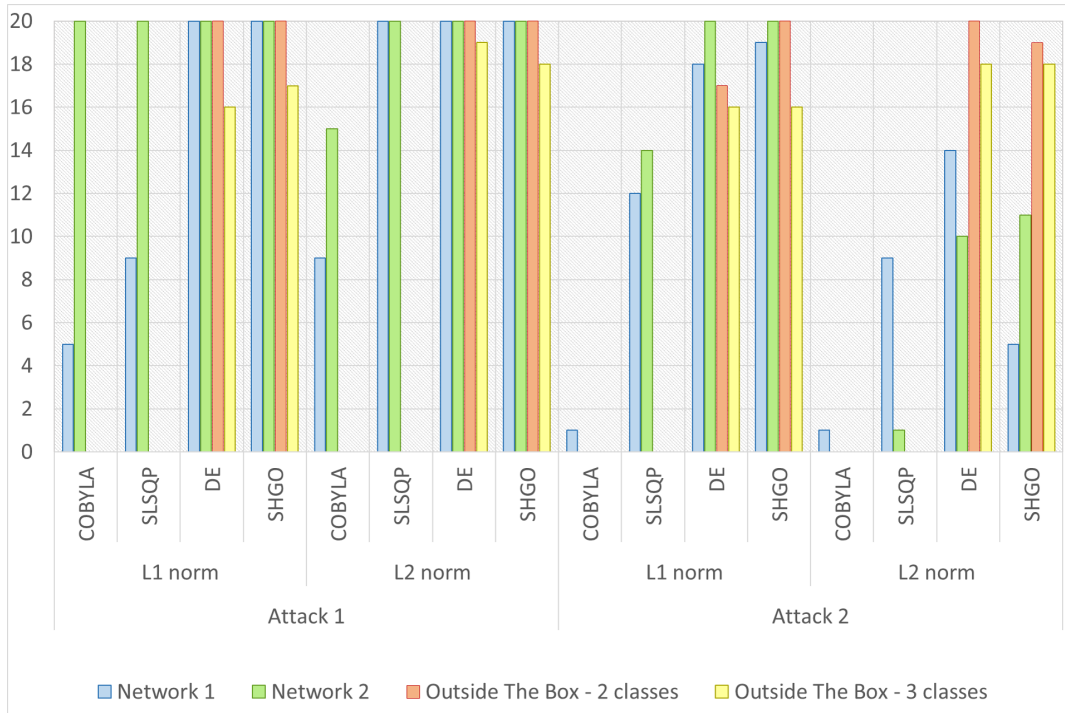


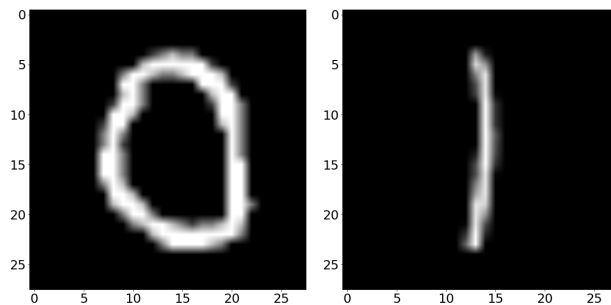
Figure 5.15: Success rate of different optimisation based methods for the four proposed attacks on different network architectures.

Overall results completely show that local optimisation methods such as SLSQP and COBYLA were unsuccessful when applied to the MNIST classifiers. We attribute this to the  $28 * 28$  dimensionality of the images. Changing many pixels results in a relatively large distance from the original point, which made the search very narrow around the given starting point. These methods converged when applied on networks 1 and 2, where the input size was 2d. So we refer dimensionality as our main trouble. Another factor that hinders the process is non-smoothness of our problem. Note that even if our objective function is linear, our constraints such as  $\text{monitor}(\mathbf{x}) = 0/1$  and  $\text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)$  are strongly non-linear, which hinders the task of the used linear solvers. Maybe when dealing with a simpler problem, dimensionality wouldn't be a trouble and

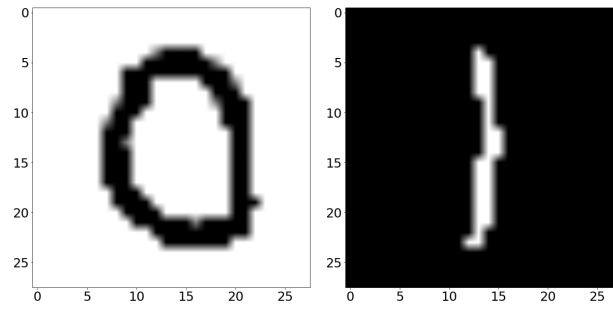


the problem could be solved by the local solvers such as COBYLA and SLSQP. We could also make local optimisation attacks more successful by giving the algorithm smarter starting points. Most algorithms guarantee convergence to local optimums if they are given a compatible initial guess. This is an easy task for networks 1 and 2 as we already know the validity and classification regions using decision boundary plots. This means that we already have track of our solution and we can easily impose a feasible start.

Global optimisation methods (DE and SHGO) however were much more successful. DE has more successful attempts than SHGO only in some few cases, but generated sample images appear very noisy making the attack easily perceived by a human observer. The reason behind this is the genetic based algorithm DE follows and the mutation of inputs that leads to obscure image results. SHGO is the best method in terms of success rate and preserving the original digit shape. Figure 5.16 illustrates examples of MNIST images before and after "Attack 1". The classifier and monitor were trained over 2 classes (0, 1) in this experiment hence these digits are detected as valid.



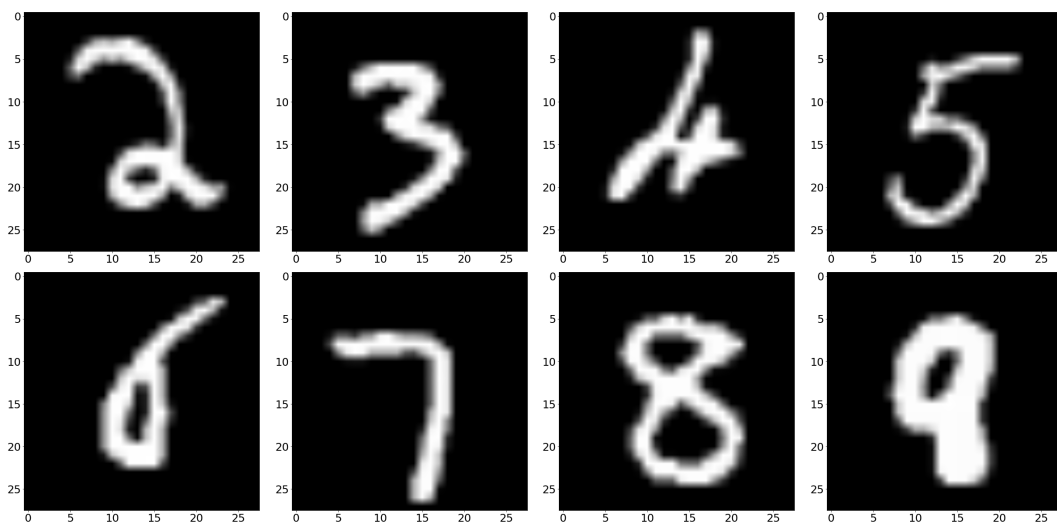
(a) Before SHGO Optimisation Attack. Samples are decided as not novel.



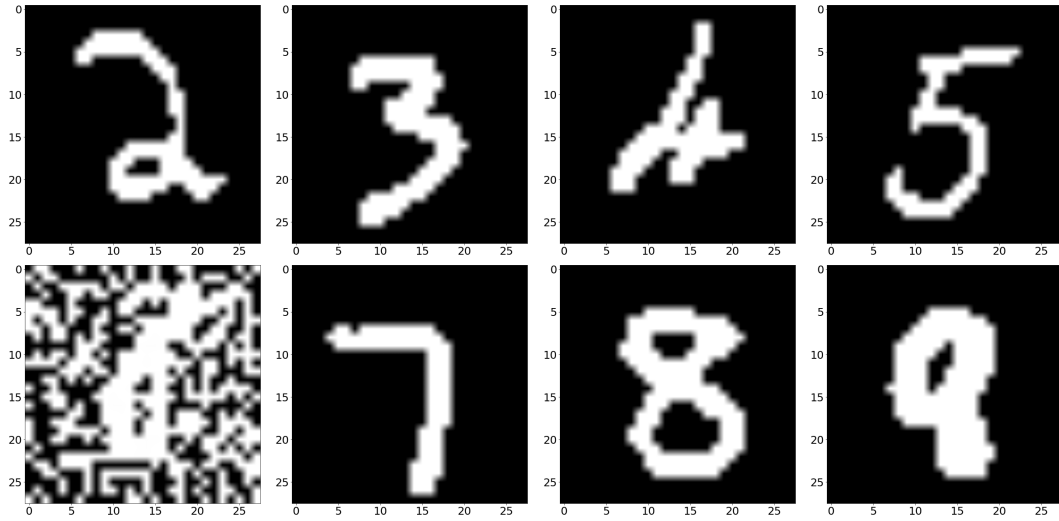
(b) After SHGO Optimisation Attack. Samples are decided as novel.

Figure 5.16: Attack 1 adversarial examples obtained by SHGO method

Figure [5.17](#) illustrates examples of MNIST images before and after "Attack 2". Since the classifier and monitor were trained over the 2 classes (0, 1) in this experiment, hence other labels (2, 3, 4, 5, 6, 7, 8 & 9) are detected as invalid.



(a) Before SHGO Optimisation Attack. Samples are decided as novel.



(b) After SHGO Optimisation Attack. Samples are decided as not novel.

Figure 5.17: Attack 2 adversarial examples obtained by SHGO method

SHGO shined since it is a derivative-free optimiser that behaves in a black box way and leverages input/output pairs. Another comparison factor is the optimiser runtime. We recorded large run-times (10 – 18 hours) for most optimisers during the experiment using a 16GB ram computer and a processor of 2.60 GHz frequency. However, SHGO converged in matter of few minutes.

# Chapter 6

## Adversarial attacks experiment

In this chapter, we show that adversarial neural network attacks are also effective when targeting the monitor of a MNIST classifier. In principle, we don't expect the monitor to detect these adversarial images as valid. A perfect novelty detector should flag adversarial samples as novel or anomalies rather than validate their wrong predictions. "Outside the box" novelty detection was not designed with the goal of detecting adversarial samples. However, we need to make sure whether such samples would be detected as novel or not.

In this experiment, We consider a neural network classifier trained over all the classes of the MNIST benchmark. We test neural network attacks over 100 samples and represent the number of successful attacks (i.e., prediction was successfully flipped) and the number of successful attacks considered as valid (i.e. monitor validates the prediction considering samples as not novel) in figure [6.1](#). Results show that most of these samples succeed into fooling the monitor and classifier. Table [6.1](#) compares the success rate of these attacks and shows a per-

turbation example. Note that in this experiment initial environment properties of the detector were used, hence a single cluster box for each class and 0 tolerance.

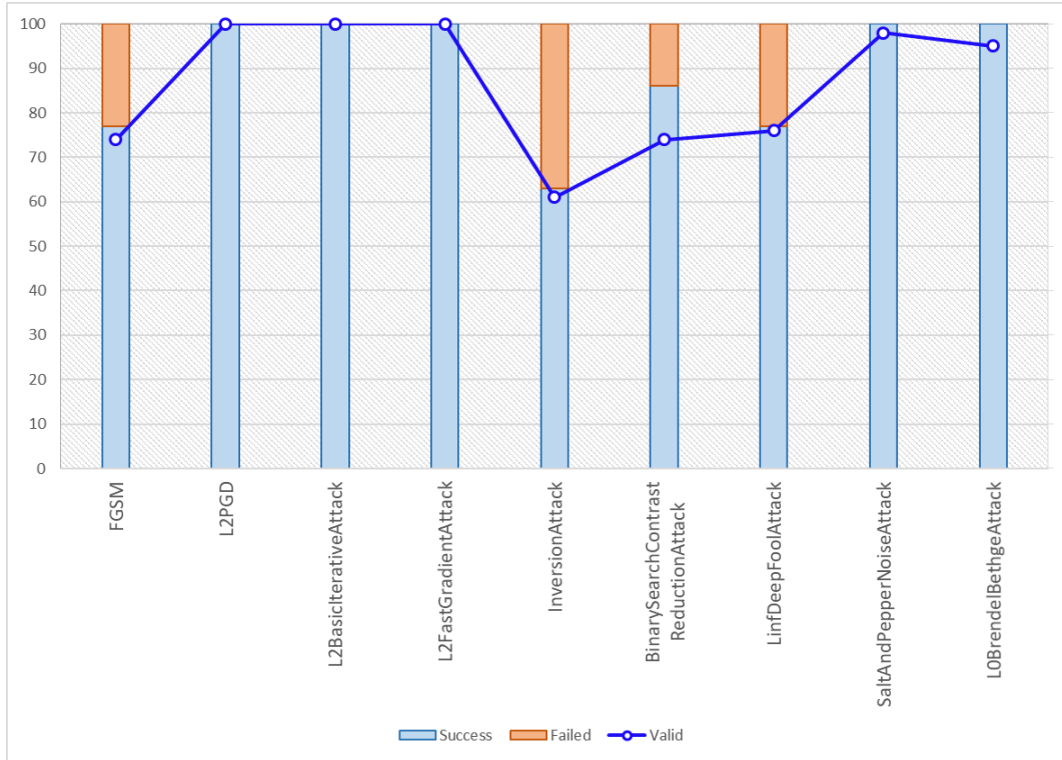
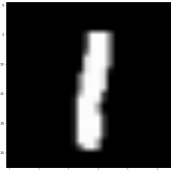
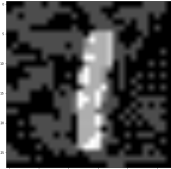



Figure 6.1: Experiment measuring the number of successful and valid attack attempts using neural network attacks

Attack	Success rate	Image
Original image	-	
FGSM	77%	
L <sub>2</sub> PGD	100%	

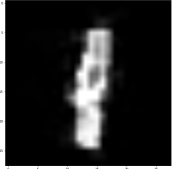
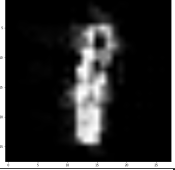
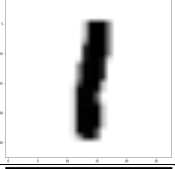
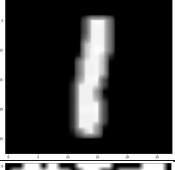

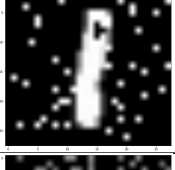

L <sub>2</sub> BasicIterativeAttack	100%	
L <sub>2</sub> FastGradientAttack	100%	
InversionAttack	63%	
BinarySearchContrastReductionAttack	86%	
L <sub>∞</sub> DeepFoolAttack	77%	
SaltAndPepperNoiseAttack	100%	
L <sub>0</sub> BrendelBethgeAttack	100%	

Table 6.1: Foolbox neural network adversarial attack result

We see that most adversarial samples are detected as valid by the monitor. These attacks fool the monitor but not completely. We need the monitor not to recognize any adversarial samples passed, hence we control some properties of "outside the box" to see its effect on validity of attack points.

## 6.1 Tuning number of clusters

We first tune the number of clusters to 1, 2 and 3. We test on same 100 samples and show the results in figure 6.2. Results show that there is no clear effect of tuning the number of clusters on the tested samples.

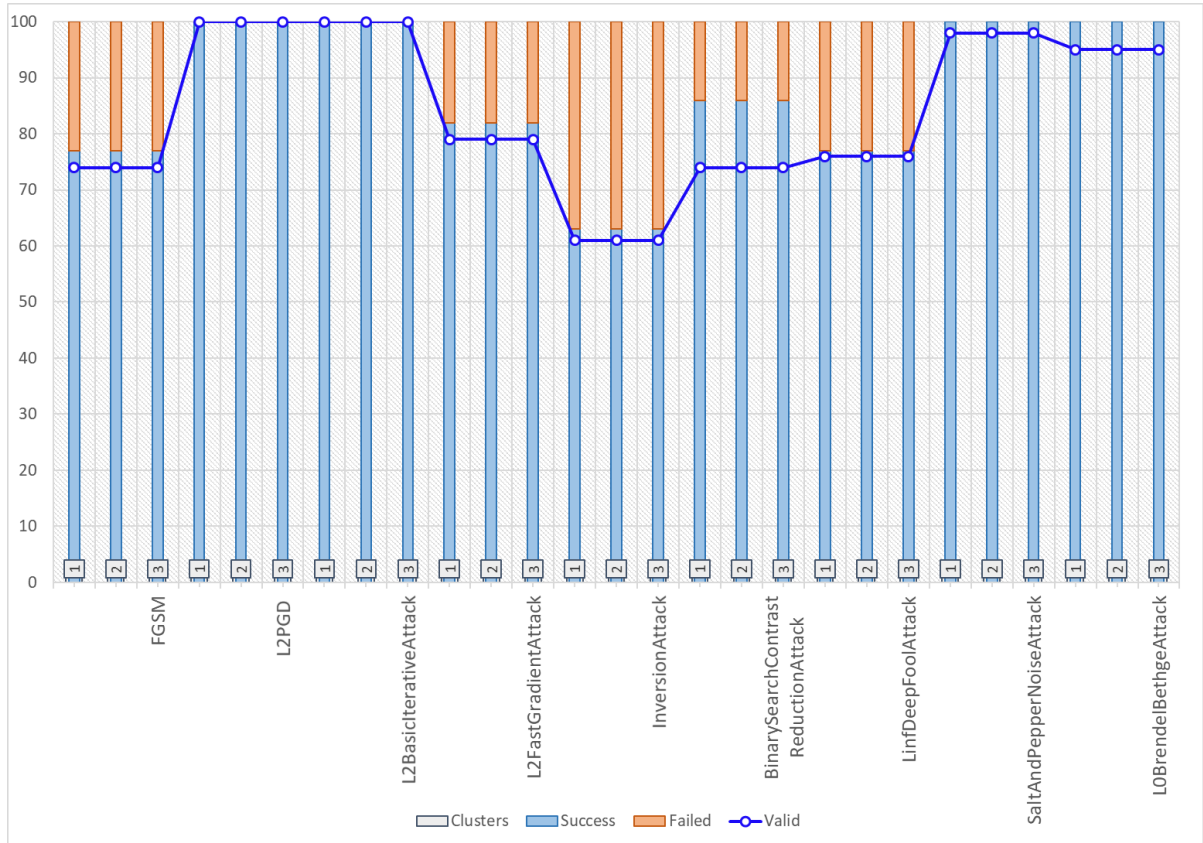


Figure 6.2: Experiment measuring the number of successful and valid attack attempts using neural network attacks while tuning different number of clusters

We analyze the effect of tuning the number of clusters visually. We show in figure 6.3 the 2d projection of inputs while we apply FGSM attack [23], using 1, 2 and 3 clusters.

From the 2d projection plot, we say that a single cluster has larger confidence

of acceptance than 2 or 3 clusters. We conclude that increasing the number of clusters per class from only one to 2 or 3 clusters per class has a positive impact on invalidating adversarial samples.

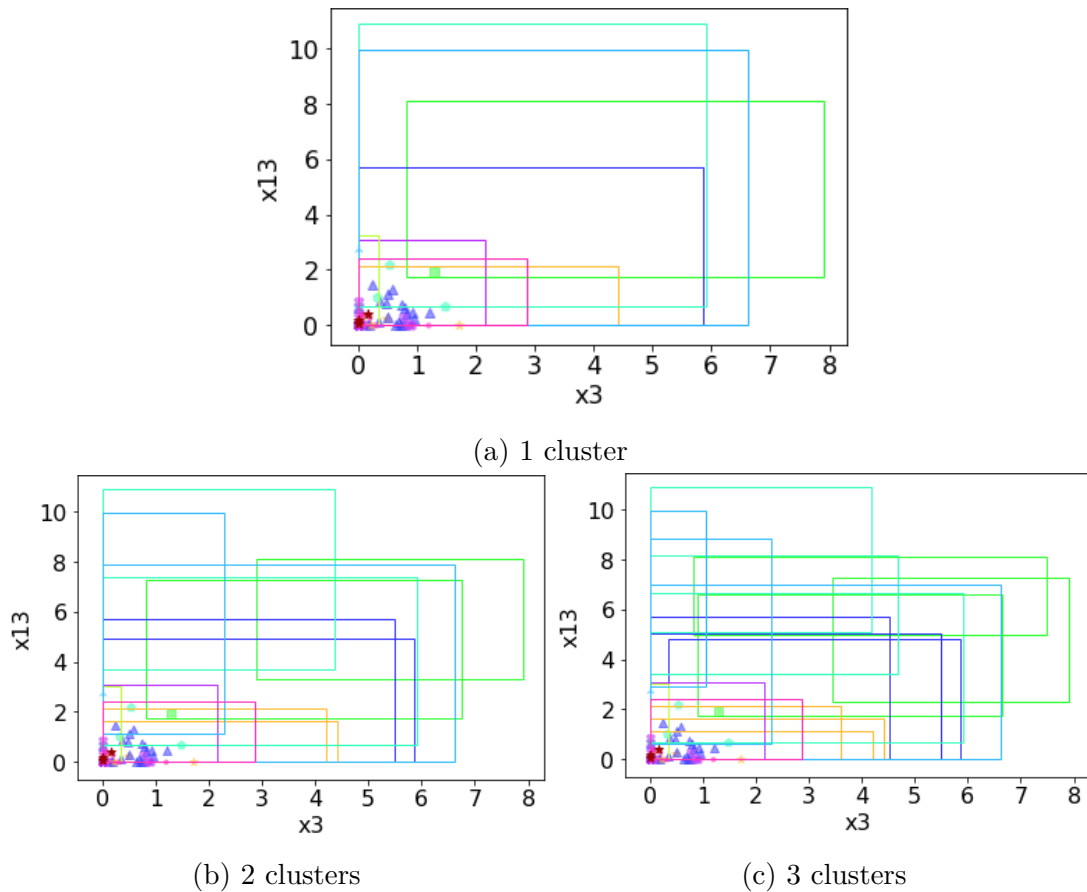


Figure 6.3: 2d projection of FGSM attack attempts while tuning different number of clusters

## 6.2 Tuning tolerance factor

We then tune box tolerance factor that is responsible for enlarging the abstraction boxes. We test on tolerance factor 0, 0.1 and 0.25. We test on same 100 samples and represent the results of attack attempts in figure [6.4](#).



The figure shows a proportional relation between validity and tolerance factor. As we increase the tolerance factor, more inputs get accepted. We clearly see this through multiple attacks.

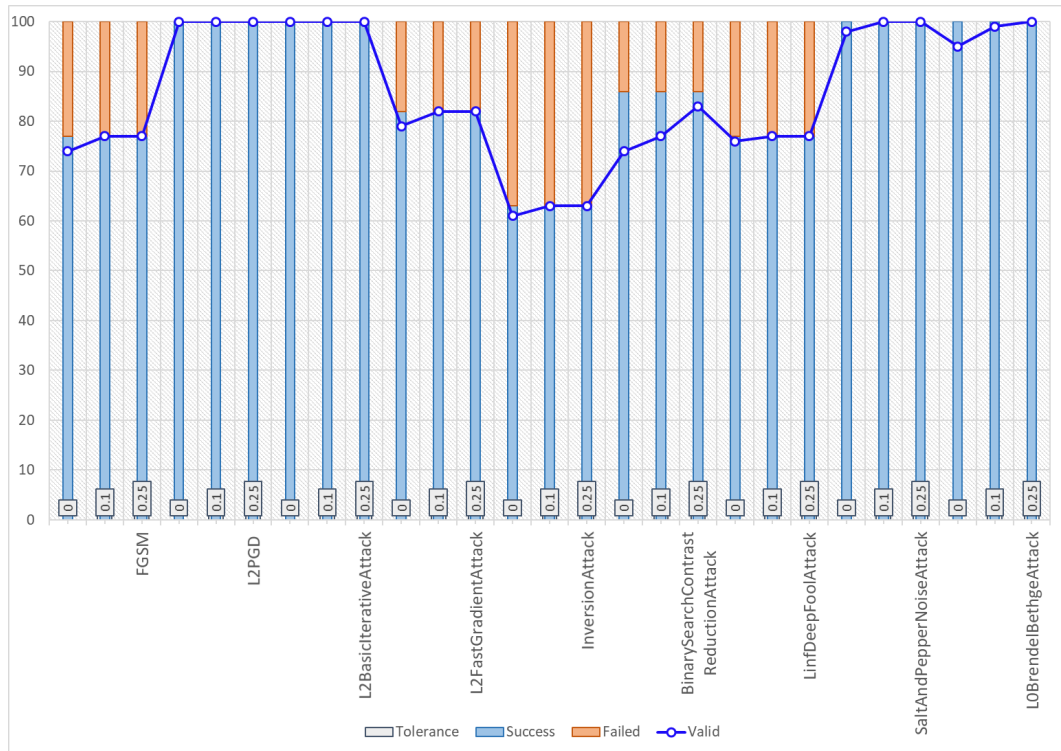


Figure 6.4: Experiment measuring the number of successful and valid attack attempts using neural network attacks while tuning different tolerance

We also analyze the effect of tuning the tolerance factor visually. We project the FGSM attack attempts and show different tolerance tolerance in figure [6.5](#). We see that more invalid inputs are being accepted and we have a wider range of confidence acceptance while increasing the tolerance factor. Increasing the tolerance factor from zero to 0.1 or 0.25 resulted in accepting more adversarial samples.

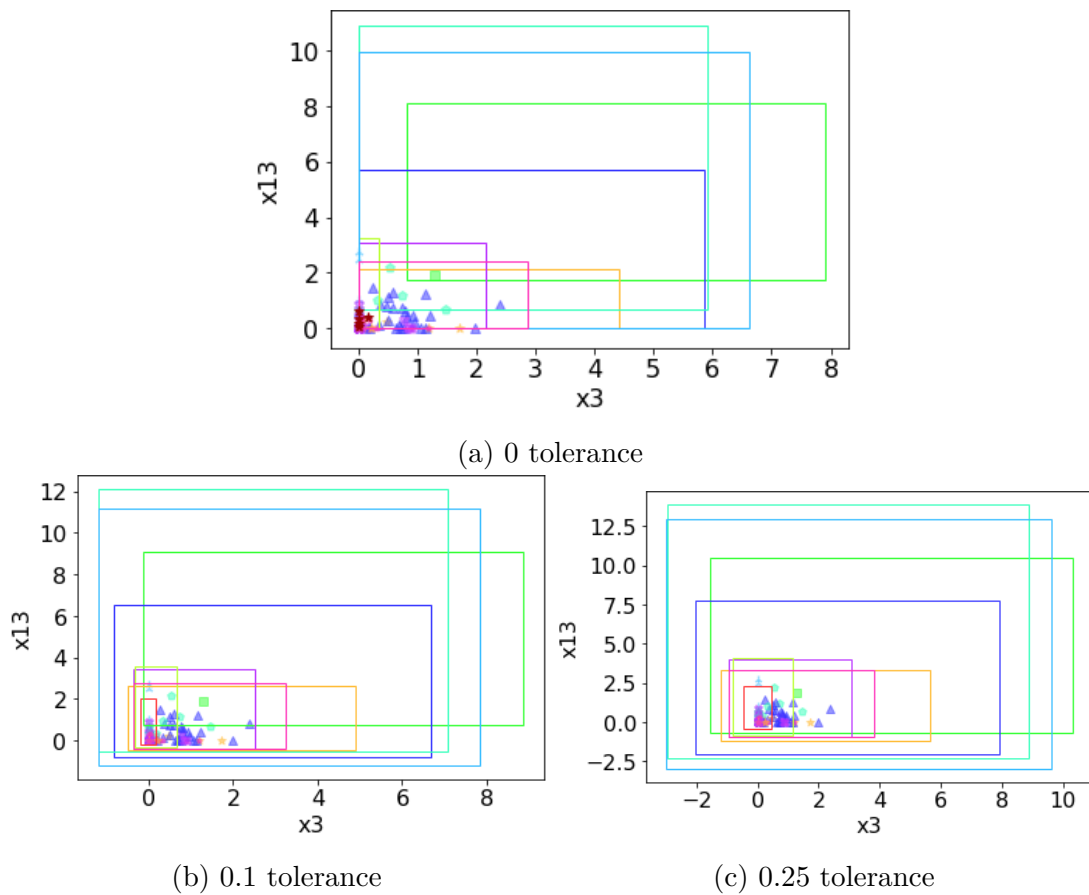


Figure 6.5: 2d projection of FGSM attack attempts while tuning different tolerance factor

However, enlarging the box by some factor, tuning the number of clusters, or trying to control any other condition in "outside the box" environment may not always lead that inputs will 100% be detected as valid by the monitor. We next improve the adversarial attacks represented in figure [6.1](#) further so that the monitor is fooled completely. All adversarial samples need to bypass the monitor.

### 6.3 Optimisation attack again!

For this experiment, we take a certain attack, for example the FGSM attack as a first trial, and run the attack on 300 samples without any optimization technique. We obtain adversarial samples that act either as valid or invalid. We then run the optimization attack, "attack 2" using  $L_1$  norm, only on invalid inputs so that they get transformed into valid. Later, all inputs are passed into monitor and get treated as valid. Note that in this example we omit the inputs where the neural network attack fails to mis-classify. Figure 6.6 shows a 2d projection of the adversarial samples as per the monitor definition. Before the optimisation attack, there are still adversarial samples marked as  $\star$  points. After the attack, all  $\star$  points disappear. The monitor now completely detects adversarial samples as valid.

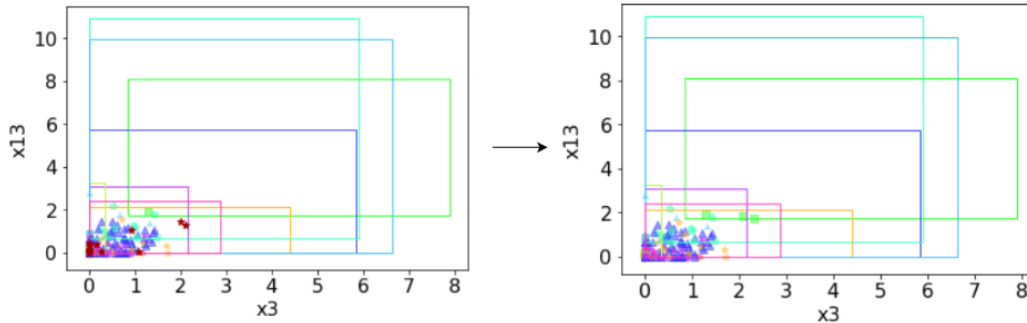


Figure 6.6: Fooling the classifier and the monitor together: FGSM adversarial samples acceptance after optimisation

We repeat same experiment but with a different attack other than FGSM for a different trial, we run the  $L_2$ FastGradient Attack on 300 samples without any optimization technique. Figure 6.7 shows a 2d projection of the adversarial samples as per the monitor definition. Before the optimisation attack, there are still

adversarial samples marked as  $\star$  points. After the attack, all  $\star$  points disappear. The monitor now completely detects adversarial samples as valid.

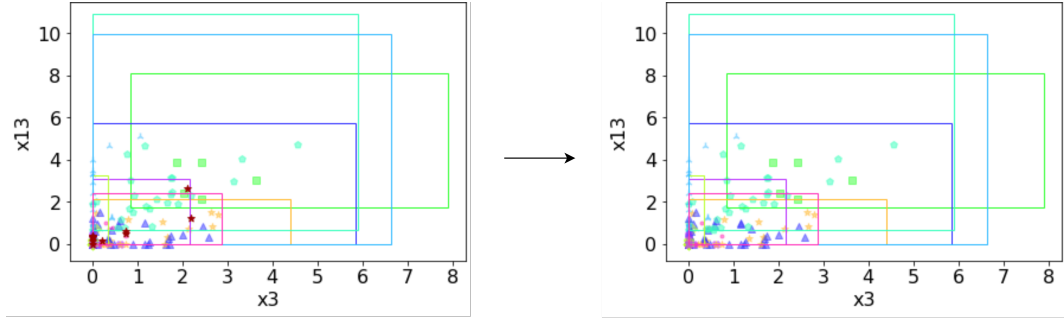
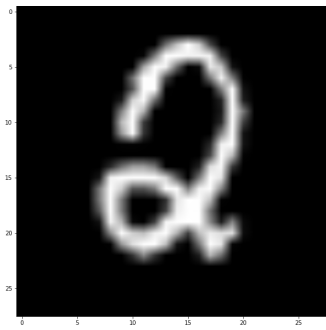


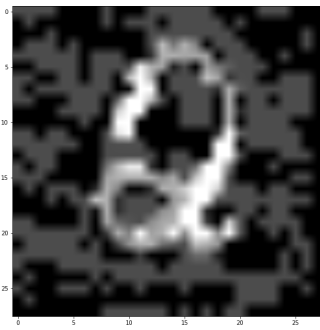
Figure 6.7: Fooling the classifier and the monitor together:  $L_2$ FastGradient adversarial samples acceptance after optimisation

We show an image example obtained by the neural network attacks in figures [6.8](#) and [6.9](#). Images in figure [6.8](#) show first the image before the FGSM attack, after FGSM attack, and after optimisation attack simulation. We also show image results of  $L_2$ FastGradientAttack while simulating adversarial attack with "attack 2" in figure [6.9](#).

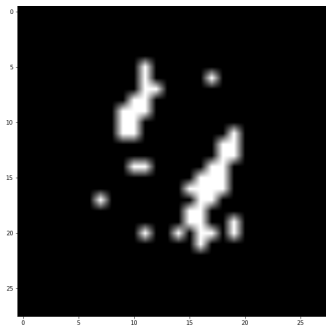
In conclusion, adversarial samples successfully detected as invalid can undergo one of our optimisation attacks to pass the monitoring test. Of course, we had to suppress the  $\text{predict}(\mathbf{x}) \neq \text{predict}(\mathbf{x}^0)$  constraint. We see that optimisation techniques succeed again to fool the monitor. SHGO method also remains a good choice to control neural network attacks.



(a) Before adversarial attack



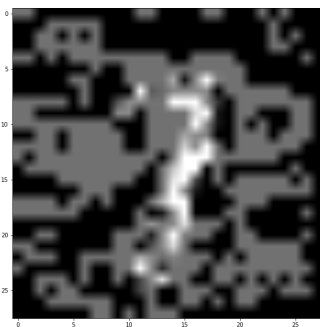
(b) After adversarial attack



(c) After adversarial + optimisation attack



(d) Before adversarial attack



(e) After adversarial attack



(f) After adversarial + optimisation attack

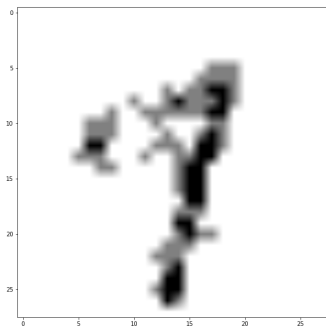
Figure 6.8: Image samples after dual attack (FGSM + attack 2)



(a) Before adversarial attack



(b) After adversarial attack



(c) After adversarial + optimisation attack

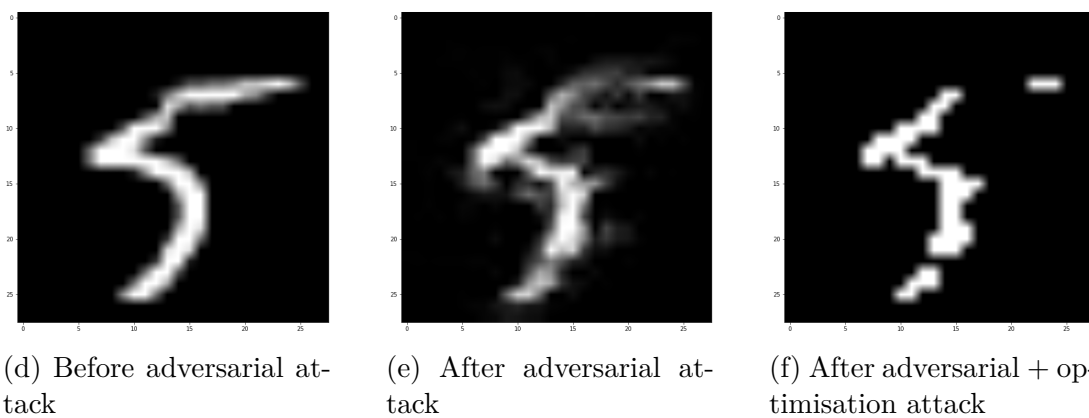


Figure 6.9: Image samples after dual attack ( $L_2$ FastGradient + attack 2)

# Chapter 7

## Defense mechanism against attacks

In this chapter, we study the effect of demonising auto-encoders against our attacks. Images produced by optimisation and neural network attacks mainly are mainly corrupted noisy images. And added noise usually changes the characteristics of the original inputs completely; thus, the first idea that comes to mind is to remove noise from adversarial examples and generate a mapping of these examples to clean examples.

Denoising auto-encoders translate input images (such as adversarial examples) into corresponding output images (aka. clean examples). We train the denoising auto-encoder via feeding it clean and noisy inputs from MNIST data-set. The denoising auto-encoder now learns to construct clean images from the noisy images. We show the impact of denoising auto-encoders against both optimisation attacks and neural network attacks.

## 7.1 Effect of denoising auto-encoders against optimisation attacks

We choose to test our defense mechanism against SHGO optimisation attack. We run the same SHGO experiment using the  $L_1$  norm difference as in chapter 5, then run the defense mechanism on top of the 20 sample images obtained by optimisation attack. Figure 7.1 shows that image reconstruction using denoising auto-encoders hinders the effect of optimisation attacks where the number of successful attack attempts drops.

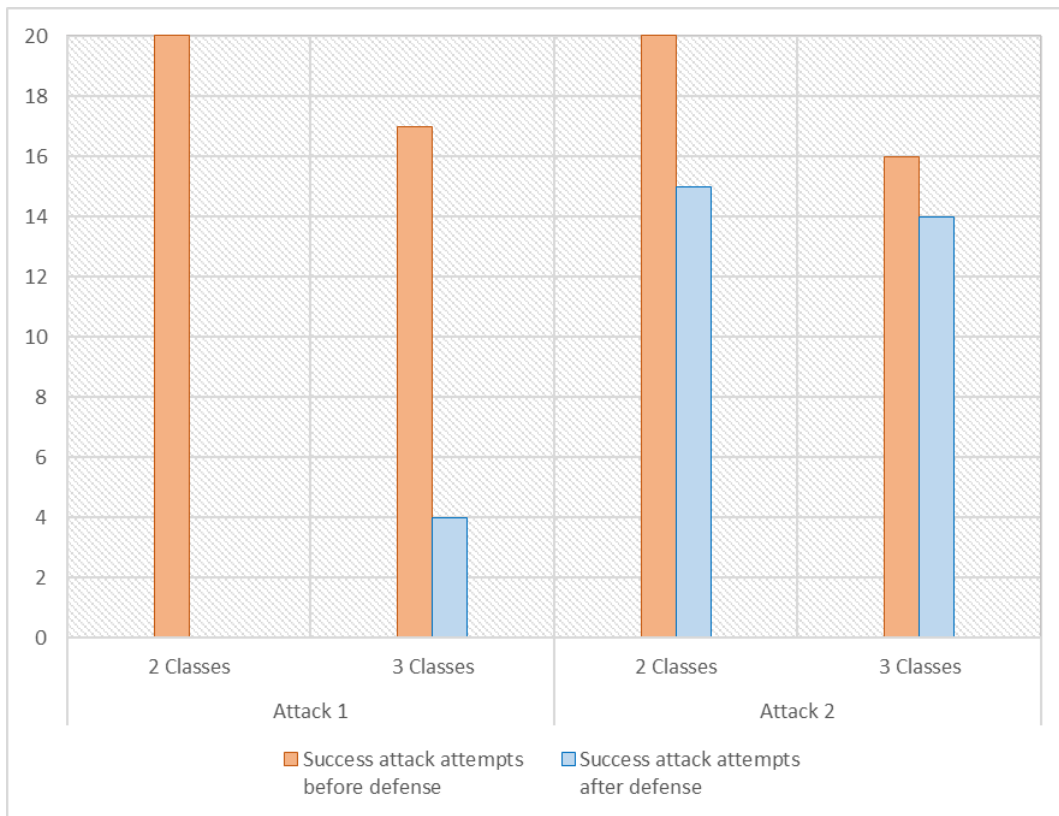


Figure 7.1: Success rate of SHGO optimisation attack after defense

We show an image example where we defend an input sample against "attack 1"



in figure 7.2. The figure shows an image before the attack 1 detected as valid, after attack 1 detected as invalid, and after reconstruction where it gets detected again as valid. We show another example where we defend against "attack 2". Figure 7.2 shows an image before attack 2 detected as invalid, after attack 2 and detected as valid, and after defense where it turns as invalid. We see that the denoising auto-encoders were able to recover the validity of the original image by reconstruction.

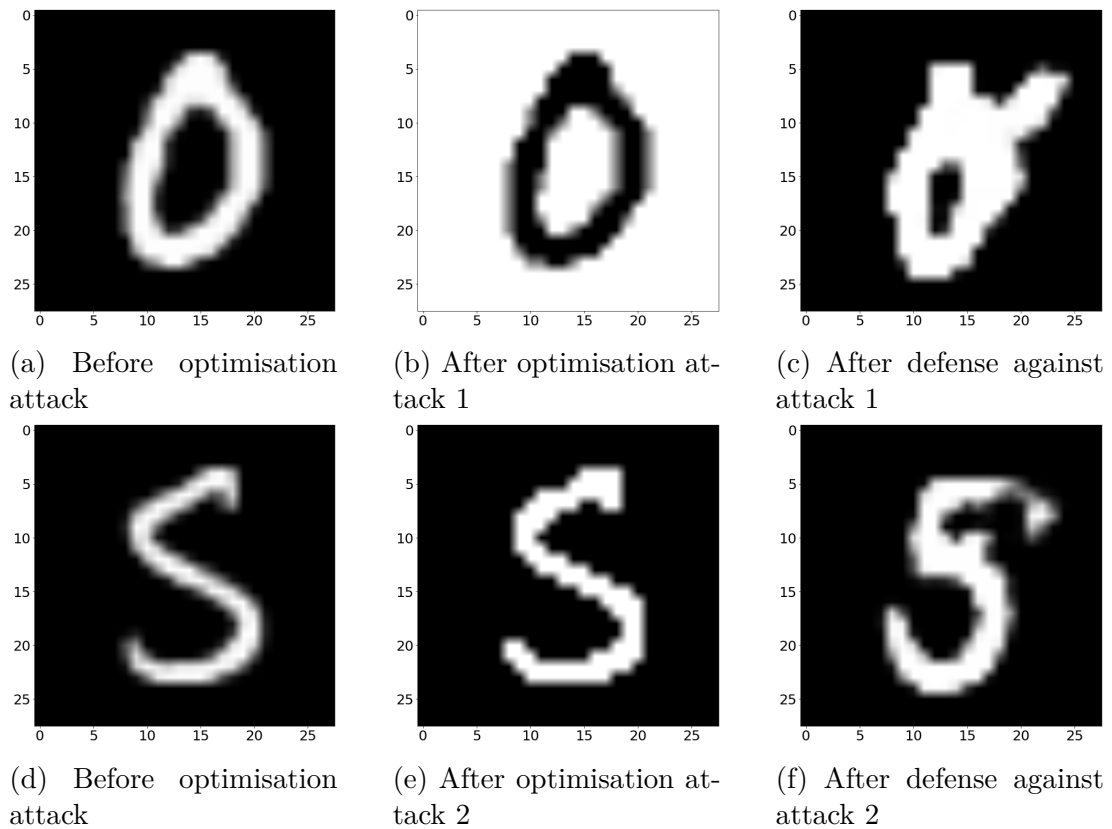


Figure 7.2: Defending an image sample against optimisation attacks

## 7.2 Effect of denoising auto-encoders against neural network attacks

We next test our defense mechanism against neural network attacks. We run again the experiment of adversarial neural network attacks in chapter 6, and run the defense mechanism on top of 100 sample images obtained by neural network attacks. Figure 7.3 shows that image reconstruction using denoising auto-encoders has a positive impact to fail adversarial attacks hence the success rate of attack attempts also drops.

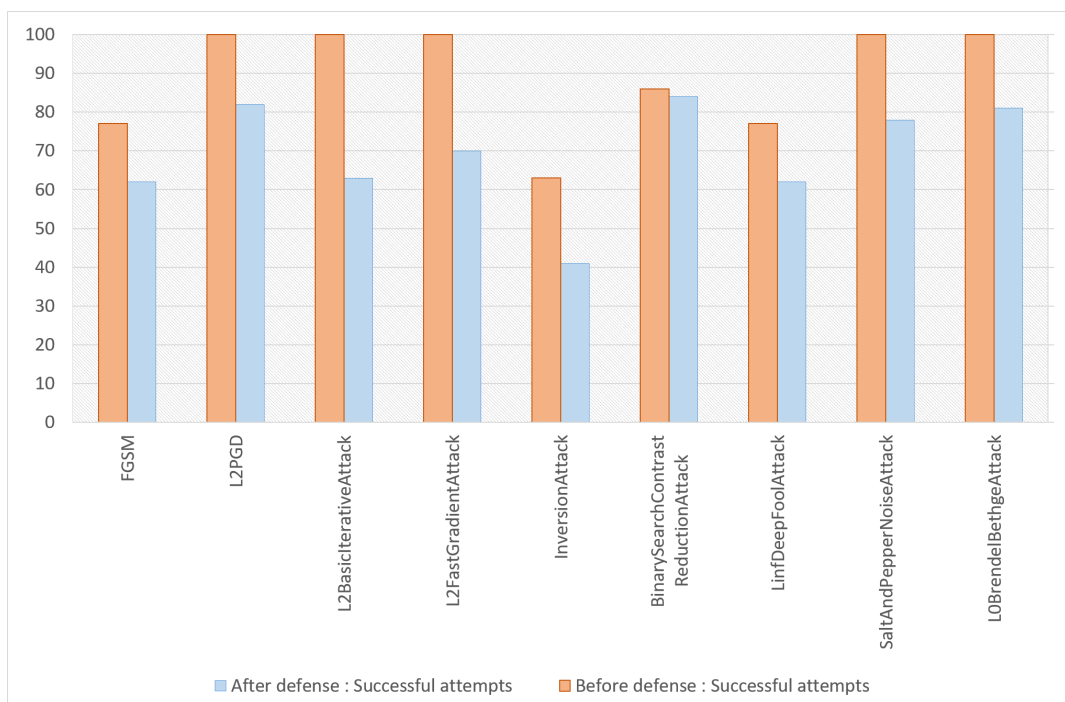
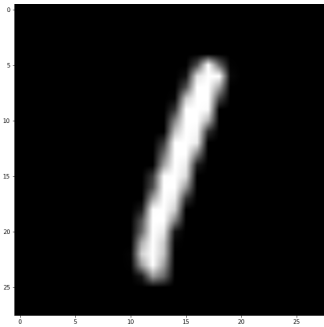


Figure 7.3: Success rate of neural network adversarial samples after defense

We show examples where we defend a testing sample against several neural network attacks. Figure 7.4 shows images before the attack belonging to class  $c$ , after the adversarial attack where the image gets mis-classified, and after recon-

struction where the image is classified again as  $c$ . We again see that denoising auto-encoders are able to recover the class of the original image by reconstruction.



(a) Before adversarial attack



(b) After FGSM adversarial attack



(c) After defense against attack



(d) Before adversarial attack



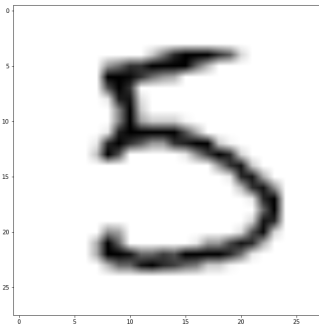
(e) After  $L_2$ FastGradient attack



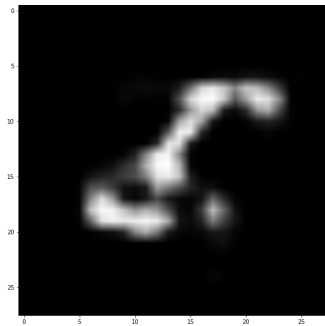
(f) After defense against attack



(g) Before adversarial attack



(h) After Inversion adversarial attack



(i) After defense against attack

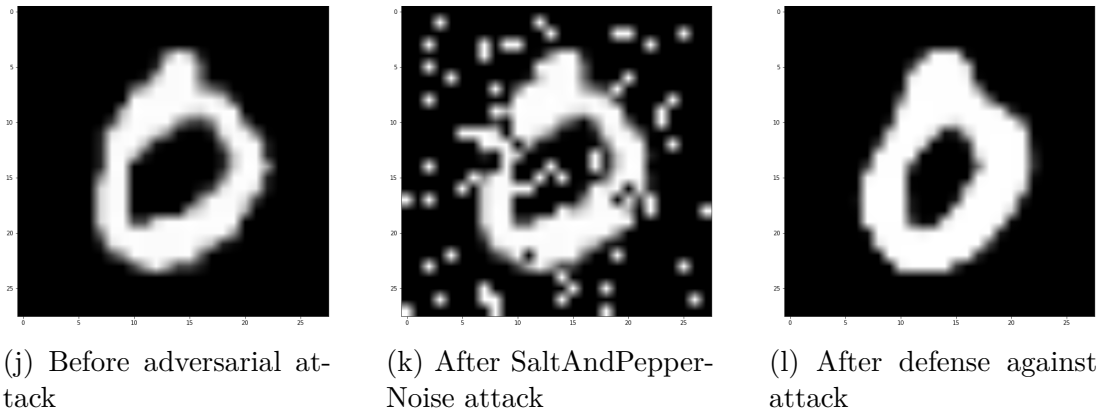


Figure 7.4: Defending an image sample against neural network attacks

We were not able to find a legitimate defense that always bypasses the monitor attacks. We did not even test "outside the box" against more sophisticated destructive attacks. Denoising auto-encoders seem a promising defense mechanism and it is still important to develop as per our experiment results. It would be a good idea to implement as a starting point in controlling adversarial attacks.

# Chapter 8

## Conclusion

In conclusion, we demonstrate that novelty detection monitors are vulnerable to fooling attacks. We were successfully able to mislead the monitor using multiple methods. We formulated optimisation problems that can be solved efficiently to find attack vectors. We also leveraged adversarial neural networks attacks from the literature to fool the classifier. We have also shown that adversarial neural network attacks combined with optimisation techniques make a deadly combo and succeed to fool both the classifier and monitor at the same time.

Novelty detection systems are always going to be vulnerable against such types of attacks as the field progresses. We must take into consideration that we will never be completely immune to attacks. Once we're exposed to a certain risk, many threat models can follow and be created.

We envision exploring more ways to help defend novelty detection against attacks. We see that denoising auto-encoders did help in defending MNIST classifiers against attacks, but not completely. The range of attacks is very expanding,

and the process of detecting and defending against attacks became very challenging. Therefore, building an effective defense mechanism whether against neural network attacks or optimisation attacks is crucial.

In summary, we hope to find a defense mechanism with better performance on our proposed attacks, and that new novelty detection techniques would consider security by design a main requirement, especially in decision-making critical systems.

# Appendix A

## Research Article

This master thesis is summarized in a research paper and was submitted to the workshop "Artificial Intelligence for Anomalies and Novelties (AI4AN 2021)" co-located with the conference "International Joint Conference on Artificial Intelligence (IJCAI 2021)": <https://sites.google.com/view/ai4an2021>. This workshop features the most recent artificial intelligence advances for recognition, detection and adaption of anomalies & novelties. The research submission was accepted for a full presentation slot. We include the reviewers comments and the research paper submitted below.

Review	Description
--------	-------------

Review 1	<p>This manuscript considers the security of novelty detection and creates attack samples to fool both the abstraction-based novelty detector and the classifier, which is a very novel perspective as far as I know. Based on optimization methods and adversarial attacks, the authors create high-quality samples to achieve successful and efficient attacks. However, there are some issues that could be addressed:</p> <ol style="list-style-type: none"> <li>1. Excessive discussion of anomaly detection/novelty detection/outlier detection in this manuscript may be unnecessary. A clear introduction of novelty detection and attacks is enough.</li> <li>2. Why choose the abstraction-based novelty detector? Given that there are many other detectors in Sec. 2, it would be great to know if the attack applies to other types of detectors as well.</li> <li>3. The solution process of the four optimization problems is not clear enough. A brief introduction or supplement might be better. Besides, Fig. 6 shows the attack samples from valid to invalid, how about the visualization of samples from invalid to valid? Readers may wonder what they look like.</li> <li>4. Last but not least, in Sec. 4.2 where 100% attack success over 300 MNIST samples with suppressing the <math>\text{predict}(x) \neq \text{predict}(x_0)</math> constraint, which leads to a puzzling problem: When the two predictions are the same, should <math>x</math> still be considered a novelty or an adversarial sample? It might be reasonable to have the monitor accept <math>x</math> in this case since <math>x</math> may be very close to <math>x_0</math> and their predictions are the same. In other words, in this case two different predictions are necessary, otherwise the attack is meaningless. So it might make more sense to report the results with two different predictions.</li> </ol>
----------	---



Review 2	<p>The authors combine two research directions, adversarial machine learning (AdvML) and anomaly detection (AD). They show possible attacks on a recent AD method by Henzinger et al., which analyses the hidden activations of neural networks for novel inputs. Generally, the topic is of interest to the AD community as misclassifications due to attacks may be costly in e.g. fraud or intrusion detection. It is a welcome step to see research in this direction.</p> <p>There are some obvious weak spots of this paper:</p> <ol style="list-style-type: none"> <li>1. The related work section is rather shallow and does not cover all relevant research directions. It may give a first overview about AD methods, but omits AdvML entirely. The authors conclude "little or none work has been conducted on the security of the aforementioned [AD] approaches" - there definitely is a lot of work on intrusion detection attacked &amp; hardened by AdvML [1,2,3,...] that should be considered.</li> <li>2. There is no discussion why the AD method by Henzinger et al. is considered as method under attack. I wish there were some hints why it may be an exemplar for AD, or how the results could be leveraged to other methods. Otherwise, it would be more beneficial to evaluate multiple methods.</li> <li>3. There is no overview about the attacks &amp; parameters used in 3.2/4.2. "In their white-box primitive version, a step is taken in the opposite direction of the gradient" (p.4) makes me believe that the authors used FGSM - but what is the step size, what is the attack budget? And above all: why using FGSM instead of state-of-the-art attacks like PGD or C&amp;W? This makes the research hard to follow &amp; reproduce.</li> </ol> <p>Furthermore, AD &amp; AdvML may run in some contradictions: is a sample pushed from anomalous to normal still anomalous? Or has it become normal as it now resides in the vicinity of normal samples? I like the discussion the authors started in their introduction about "[novelty detection] is parameterized by both the data and the classifier" (p.1). In my opinion, this demands at least a check in the evaluation whether the adversarial examples may be shifted towards normal classes. Similiar intuition goes to chapter 4.2: "a perfect novelty detector should flag adversarial samples as novel" (p.6) - but should it? I see that e.g. a F-MNIST sample should never be detected as normal for an novelty detector trained on MNIST - but I doubt that AdvML will ever push an MNIST sample so far away that it will not be close to an MNIST sample without losing any context features.</p> <p>I cannot endorse the paper in its current form, but would like to encourage the authors to follow the research they have started. The overlap of AD &amp; AdvML is interesting and demands new thoughts and thorough studies.</p> <p>[1] "Analyzing Adversarial Attacks against Deep Learning for Intrusion Detection in IoT Networks" by Ibitoye et al. [2] "Investigating Adversarial Attacks against Network Intrusion Detection Systems in SDNs" by Aiken and Scott-Hayward [3] "Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT" by Papadopoulos et al.</p>
----------	--

Review 3	<p>I recommend weak accept for this paper. The paper is centered on an interesting area, specifically on attacks against novelty detectors (in this case, the authors mean the problem of getting a neural network to predict when an input sample is from a novel/unseen class from the ones used in training). The authors propose a reasonable strategy to attack one such novelty detection approach which uses bounding boxes on layers, via an optimization problem involving small perturbations to the input. I think the main gripe with this work is that it misses a lot of related work and citations for relevant and pre-existing approaches, which is especially crucial since this strategy of attack is quite common in prior literature, though the application domain here seems new. I encourage the authors to revise appropriately to better contextualize their contributions.</p> <p>Comments:</p> <ol style="list-style-type: none"> <li>1. "However, once trained over a finite set of classes, a deep learning model does not have the power to say that a given input does not belong to any of the classes and simply cannot be linked" – this claim seems a bit strong. It seems this would be achievable with an appropriate loss choice and heuristic. In fact, the next line says "...has been tackled in many ways in the literature" implying the problem has been worked on and solutions do exist.</li> <li>2. The setting the authors discuss seems quite similar to zero shot learning (ZSL) in the ML domain. Adding clarifications/distinguishing factors, as well as some related work on this space could more appropriately contextualize the work.</li> <li>3. This type of "imperceptible" attack (i.e. perturbation within a small epsilon-ball/budget to maximize confusion of a classifier) has been widely studied (e.g. <a href="https://www.kdd.org/kdd2018/accepted-papers/view/adversarial-attacks-on-neural-networks-for-graph-data">https://www.kdd.org/kdd2018/accepted-papers/view/adversarial-attacks-on-neural-networks-for-graph-data</a> is one example which does this, but there are many others). It would be useful to contextualize the contribution in this work with respect to these prior works (e.g. adding relevant citations after mentioning your proposal to formulate each attack as an optimization problem in Sec 3).</li> <li>4. Figure 4 caption could be better written (it's currently all lower caps and not well-explained).</li> <li>5. Some examples of the discussed "very noisy" adversarial generated samples from methods like DE/SHGO could be interesting from a reader perspective (you might be able to include these by using less space for other figures).</li> <li>6. Figure 7 axes are unmeaningful/hard to interpret.</li> <li>7. FGSM is introduced without any citation or explanation on page 6.</li> </ol>
----------	---

Table A.1: Reviewers comments

# Hack The Box: Fooling Deep Learning Abstraction-Based Monitors

Sara Hajj Ibrahim<sup>1</sup>, Mohamed Nassar<sup>2</sup>,

Department of Computer Science  
Faculty of Arts and Sciences  
American University of Beirut (AUB)  
sih11@mail.aub.edu, mn115@aub.edu.lb

## Abstract

Deep learning is a type of machine learning that adapts a deep hierarchy of concepts. Deep learning classifiers link the most basic version of concepts at the input layer to the most abstract version of concepts at the output layer, also known as a class or label. However, once trained over a finite set of classes, a deep learning model does not have the power to say that a given input does not belong to any of the classes and simply cannot be linked. Correctly invalidating the prediction of unrelated classes is a challenging problem that has been tackled in many ways in the literature. Novelty detection gives deep learning the ability to output "do not know" for novel/unseen classes. Still, no attention has been given to security aspects of novelty detection. In this paper, we consider the case study of abstraction-based novelty detection and show that it is not robust against adversarial samples. Moreover, we show the feasibility of crafting adversarial samples that fool the deep learning classifier and bypass the novelty detection monitoring at the same time. In other words, these monitoring boxes are hackable. We demonstrate that novelty detection itself ends up as an attack surface.

## 1 Introduction

Machine learning algorithms are excellent at analyzing data and finding interesting patterns. However, they give up to the so-called dimensionality curse. It was shown that deep learning bypasses the traditional machine learning algorithms in most learning tasks in the literature [Nassar, 2020]. While deep learning yields remarkable results in the field of raw data representation and classification, it suddenly becomes sub-optimal when explaining decisions or recognizing a novel class of input. Supervised deep neural networks never say "I don't know", they can be just less or more confident about an outcome or decision. The necessity of monitoring deep learning for novelty and anomaly detection is directly visible.

Novelty detection can play a significant role and be leveraged for monitoring and discovering new classes that were unseen during training time. However, most work on novelty detection give no attention to its security aspects. In this

paper, we show that from a security perspective, novelty detection can be easily attacked and may augment the attack surface of deep learning based systems.

We distinguish between three interrelated and very close concepts, namely anomaly detection, outlier detection and novelty detection. These terms are sometimes interchangeably used in literature, but we suggest that they mean different things and it is time to give each an appropriate definition. In our terminology, anomaly detection stems from unsupervised one-class modeling or supervised binary classification into normal and abnormal. Outlier detection stems from unsupervised learning and consists on finding points that likely not belong to any clusters found in unlabeled data. Novelty detection is the process of distinguishing between data inputs belonging to one of the classes encountered during the training time and data inputs belonging to classes that are previously unseen. It is different than outlier detection since data have labels. It is also different from anomaly detection since it is parameterized by both the data and the classifier whereas anomaly detection is usually solely parameterized by the data. Novelty detection in deep learning is a new and active research area. Abstraction-based novelty detection is one of the main proposed approaches. This approach summarizes training input and intermediate data representations into statistical constructs that makes it easy to detect novelty at testing stage.

A white-box abstraction-based novelty detection method is proposed in [Henzinger *et al.*, 2020]. A monitor takes a one-by-one decision on each testing sample and identifies it as either valid (i.e. the classifier prediction is correct on assigning the sample to one the training classes), or invalid (i.e. the classifier prediction is rejected). The monitor decision is based on verifying whether a special representation of the sample falls within one of the boxes constructed during training or outside these boxes. This special representation is based on values taken from internal neural nodes at hidden layers. Each class has its own box or set of boxes. We take these monitors as a case-study in this paper and show that: (1) these monitors are not efficient when adversarial testing samples are presented, and (2) these monitors can themselves be attacked by appending the adversarial generation process with new constraints. In other words, these boxes are hackable.

The remaining of this paper is organized as follows: Section 2 summarizes our terminology and literature review. Our

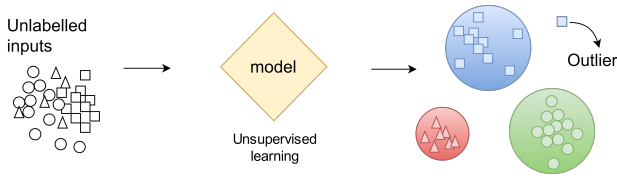


Figure 1: An example of an outlier detection system

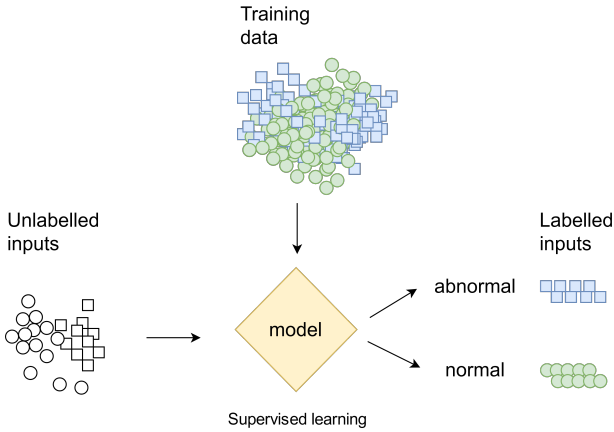


Figure 2: An example of an anomaly detection system

attack methodology is presented in section 3. Section 4 evaluates our experiments and findings. Finally Section 5 concludes the paper and sketches future work.

## 2 Background & Literature Review

First let's be more precise about what is novelty detection and how it differs from outlier and anomaly detection. An outlier is an "Observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism" [Hawkins *et al.*, 1980]. Outlier detection is the process of coining observations that significantly deviate from the majority of data. Unsupervised algorithms extract statistical information indicating how unlikely a certain observation is to occur, for example, finding a point deviating far from the statistical means of other points as illustrated in Figure 1.

An anomaly is a special case of outliers which is usually tied to special information or reasons [Aggarwal, 2015]. Anomalies indicate significant and rare events that may prompt critical actions in a wide range of application domains [Ahmed *et al.*, 2016]. Anomaly detection may require labeled data and employ supervised algorithms as illustrated in Figure 2. For example, we consider the problem of malware/benign classification as a form of anomaly detection.

Novelty detection is the process of identifying inputs that belong to unknown classes that were not provided during training time. Consider a supervised learner having  $c$  classes at training time but  $c + u$  classes appear at testing time. The goal of novelty detection is to invalidate the output of the classification when samples from the  $u$  classes are presented. Novelty detection is different than the previously described

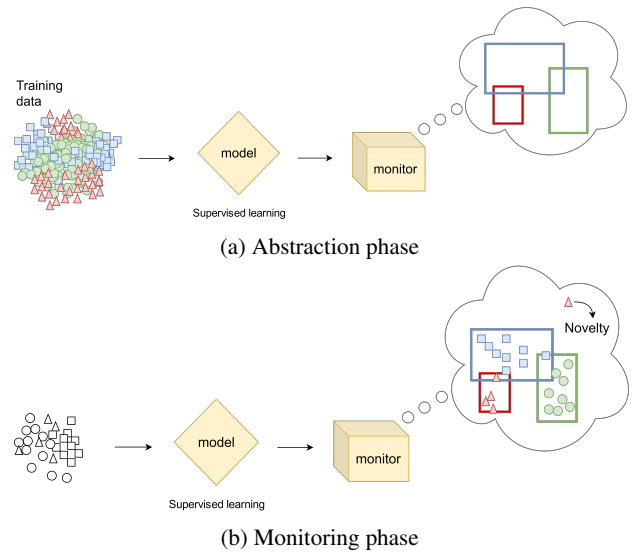


Figure 3: An example of novelty detection system based on box abstractions

anomaly and outlier detection for two main reasons: (1) training data have labels, and (2) the learner itself is an input to the detector algorithm.

Novelty detection can be achieved in white-box mode by taking the model obtained after training and building a monitor on top of it. The monitor fingerprints the behaviour of the model when training data are presented. For the special case of hidden nodes given by forward-propagating the training samples. The monitored values are abstracted into statistical constructs. Later, outlier detection flag inputs having fingerprints that largely deviate from these constructs. In other words, this approach transforms the novelty detection problem into an outlier detection problem by projecting the data into a different hyper-dimensional space. This projection is parameterized by the neural network model itself. Different types of abstractions are proposed in [Henzinger *et al.*, 2020] before evaluating box abstractions in particular. Figure 3 shows an example of a box abstraction-based novelty detection system.

Next we summarize the main approaches for addressing novelty detection in deep learning from the literature.

**Distance based methods** These methods compute novelty values or confidence scores based on distance metric functions. In [Mandelbaum and Weinshall, 2017] the data are first embedded as derived from the penultimate layer of the neural network. The confidence score is based on the estimation of local density. Local density at a point is estimated based on the Euclidean distance in the embedded space between the point and its  $k$  nearest neighbors in the training set. A similar approach based on learning a local model around a test sample is proposed on [Bodesheim *et al.*, 2015] for Multi-class novelty detection tasks in image recognition problems.

**Statistical Based Methods** Novelties are caused by differences in data distributions at training and prediction

time. Some of these methods require sampling the distribution at run-time or an online adaptation of classifiers. In [Pidhorskyi *et al.*, 2018] the underlying structure of the inlier distribution of the training data is captured. The novelty is detected by the means of a hypothesis test or by computing a novelty probability value.

**Auto-Encoding and Reconstruction Based Methods** One way to proceed is to train a deep encoder-decoder network that outputs a reconstruction error for each sample. The error is used to either compute a novelty score or to train a one-class classifier. [Pidhorskyi *et al.*, 2018] also uses an auto-encoder network but to derive a linearized manifold representation of the training data. The manifold representation helps compute a novelty probability that represents how likely it is that a sample was generated by the inlier distribution. This is why we consider it as a statistical based method in the same time.

[Domingues *et al.*, 2018] introduces an unsupervised model for novelty detection based on Deep Gaussian Process Auto-Encoders (DGP-AE). The proposed auto-encoder is trained by approximating the DGP layers using random feature expansions, and by performing stochastic variational inference on the resulting approximate model. Their work can be categorized under anomaly detection in our terminology.

**Bayesian methods** These methods use Bayesian formalism to detect anomalies and new classes in addition to classification [Roberts *et al.*, 2019]. The basic idea is to add a "dummy" class at the root node. The class is considered under-represented in the training set. The classifier gives a strong a posteriori of being "dummy" for unseen instances.

**Abstraction based methods** These methods consider a finite set of vectors  $X$ , and construct a set  $Y$  that generalizes  $X$  to infinitely many elements and has a simple representation that is easy to manipulate and answer queries for. Examples of these methods are ball-abstraction such as one-class support vector machines, one-class neural networks [Chalapathy *et al.*, 2018] and box-abstraction [Henzinger *et al.*, 2020].

However, little or none work has been conducted on the security of the aforementioned approaches either in the presence of adversarial samples for fooling the classifier or against especially crafted samples for fooling the novelty detector and the classifier at the same time. For our study, we consider the use-case of "outside the box" [Henzinger *et al.*, 2020] and analyse its security in different ways.

### 3 Methodology

In [Henzinger *et al.*, 2020], constructing an abstraction at layer  $l$  of the monitored network for class  $y$  works as follows:

1. Collect outputs at layer  $l$  for inputs of class  $y$
2. Divide collected vectors into clusters
3. Construct an abstraction for vectors in each cluster, e.g. an enclosing box.

Monitoring at layer  $l$  works as follows:

1. Predict class of input  $x$
2. Collect output at layer  $l$  into a vector  $v$
3. Check if any of the abstraction of the predicted class contains  $v$
4. The prediction is rejected if the check returns empty.

We distinguish two types of attacks against this schema:

**Attack 1 - from valid to invalid** Consider an input  $x$  which would normally be identified as valid by the monitor and belongs to one of the training classes. This attack modifies  $x$  in a slight and unperceivable way to make it get rejected by the monitor. As an example of application of this attack, we would imagine a denial of service for a legitimate user in a face recognition system.

**Attack 2 - from invalid to valid** Consider an input  $x$  which would normally be rejected by the monitor as not belonging to any of the classes seen during training. This attack modifies  $x$  in a slight and unperceivable way to make it get accepted by the monitor. The attack can be targeting a preset prediction, or just going with any prediction output by the neural network. As an example of application, we would imagine letting go an intruder in a face recognition system. The intruder is identified as any of the legitimate white-list users.

In terms of implementation, we propose to formulate each attack as an optimisation problem that can be solved by an iterative optimisation algorithm. We also experiment with off-the-shelf adversarial attacks against neural networks and assess their efficiency, as well as augmenting them by an optimisation component to target a specific attack and make them more efficient. We detail these two approaches next.

#### 3.1 Optimisation based attacks

##### Attack 1: from valid to invalid

Consider a neural network that is trained over only two classes, where each class is represented by the monitor under one box. As shown in figure 4(a), we push the images of valid points, as represented by the monitor, from both classes  $\blacksquare$  and  $\bullet$  to fall outside their boxes and therefore be marked as novel exactly as for the  $\star$  points.

More generally, consider a point  $x^0$  such as  $\text{monitor}(x^0, c) = 1$  (accept as class  $c$ ), our goal is to find a point  $x$  as close as possible to  $x^0$  such that  $\text{monitor}(x, c) = 0$  (reject). For measuring the distance between the two points we either use the  $L_1$  norm or the  $L_2$  norm. In addition, we require that  $x$  preserves the same prediction as of  $x^0$  via the monitored network:  $\text{predict}(x) = \text{predict}(x^0) = c$ .

In case of the  $L_1$  norm, we replace the non-differentiable objective function by a differentiable one through introducing a vector  $z$  having the same dimension as  $x$ . We get a linear

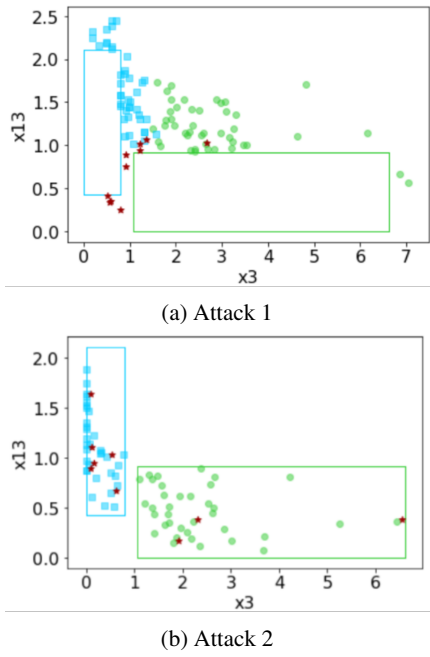


Figure 4: optimisation attack generalization

programming problem as follows:

$$\begin{aligned}
 & \text{Minimize} && \sum_{i=1}^n z_i \\
 & \text{Subject to} && z_i + (x_i - x_i^0) \geq 0 \\
 & && z_i - (x_i - x_i^0) \geq 0 \\
 & && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 0 \\
 & && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
 \end{aligned} \tag{1}$$

We next use the  $L_2$  norm ( $\|\mathbf{x} - \mathbf{x}^0\|_2$ ) to formulate a second optimisation problem. The constraints for  $L_2$  are same as for  $L_1$  except that  $L_2$  norm is already a differential objective function and can be directly tuned by a linear solver. The  $L_2$  norm attack focuses on minimizing the square root of the sum of squared differences of  $(\mathbf{x}^0)$  and  $(\mathbf{x})$  elements. We then get a linear programming problem as follows:

$$\begin{aligned}
 & \text{Minimize} && \sqrt{\sum_{i=1}^n |x_i - x_i^0|^2} \\
 & \text{Subject to} && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 0 \\
 & && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
 \end{aligned} \tag{2}$$

### Attack 2: from invalid to valid

In this attack, we push the images of invalid points  $\star$ , as represented by the monitor, towards the boxes representing legitimate classes. The points will be marked by the monitor as valid if, at the same time, the neural network prediction matches the box owner, either as  $\blacksquare$  or  $\bullet$ . The 2d projection of the monitor representation for a binary classifier is shown in figure 4(b).

Given a point  $\mathbf{x}^0$  where  $\text{monitor}(\mathbf{x}^0, c) = 0$  (reject), our goal is to find a point  $\mathbf{x}$  as close as possible to  $\mathbf{x}^0$  such that  $\text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1$  (accept as same class of  $\mathbf{x}^0$ ). For measuring the distance between the two points we choose the  $L_1$  norm or the  $L_2$  norm.

For the  $L_1$  norm, we formulate the following problem:

$$\begin{aligned}
 & \text{Minimize} && \sum_{i=1}^n z_i \\
 & \text{Subject to} && z_i + (x_i - x_i^0) \geq 0 \\
 & && z_i - (x_i - x_i^0) \geq 0 \\
 & && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1 \\
 & && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
 \end{aligned} \tag{3}$$

For the  $L_2$  norm, we formulate the following problem:

$$\begin{aligned}
 & \text{Minimize} && \sqrt{\sum_{i=1}^n |x_i - x_i^0|^2} \\
 & \text{Subject to} && \text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1 \\
 & && \text{predict}(\mathbf{x}) = \text{predict}(\mathbf{x}^0)
 \end{aligned} \tag{4}$$

The four formulated problems can be efficiently solved by constrained optimisation numerical methods that are either local such as SLSQP [Kraft and others, 1988] and COBYLA [Powell, 1994] or global such as Differential Evolution (DE) [Price, 2013] and SHGO [Endres *et al.*, 2018]. We used implementations from the SciPy library [Virtanen *et al.*, 2020] to show case these attacks as will be detailed later in section 4.

## 3.2 Adversarial attacks against neural networks

Another idea is to use known adversarial attacks against neural networks as a starting point for attacking the monitor. Adversarial neural network attacks aim at changing the prediction of an input  $\mathbf{x}^0$  when replaced by a close point  $\mathbf{x}$  as of  $\text{predict}(\mathbf{x}) \neq \text{predict}(\mathbf{x}^0)$ . In their white-box primitive version, a step is taken in the opposite direction of the gradient of the objective function at the point  $\mathbf{x}^0$ . It is interesting to check whether adversarial samples would be detected as novel by the novelty monitor. In case these samples are not, we may consolidate by optimisation based methods to make them pass as valid points. In other words, we can take an adversarial sample as a starting point for our search for an attack against both the neural network and the monitor. Our goal here is to find a point  $\mathbf{x}$  very close to  $\mathbf{x}^0$  such that  $\text{predict}(\mathbf{x}) \neq \text{predict}(\mathbf{x}^0)$  and  $\text{monitor}(\mathbf{x}, \text{predict}(\mathbf{x})) = 1$  (accept).

We use the implementation of adversarial attacks from the Foolbox library [Rauber *et al.*, 2020]. We assess the performance of an abstraction based monitor to flag adversarial samples as novel. We study the effect of tuning "outside the box" parameters to enhance the robustness of the monitor. In a second time, we use optimisation techniques to force adversarial samples to bypass the monitor.

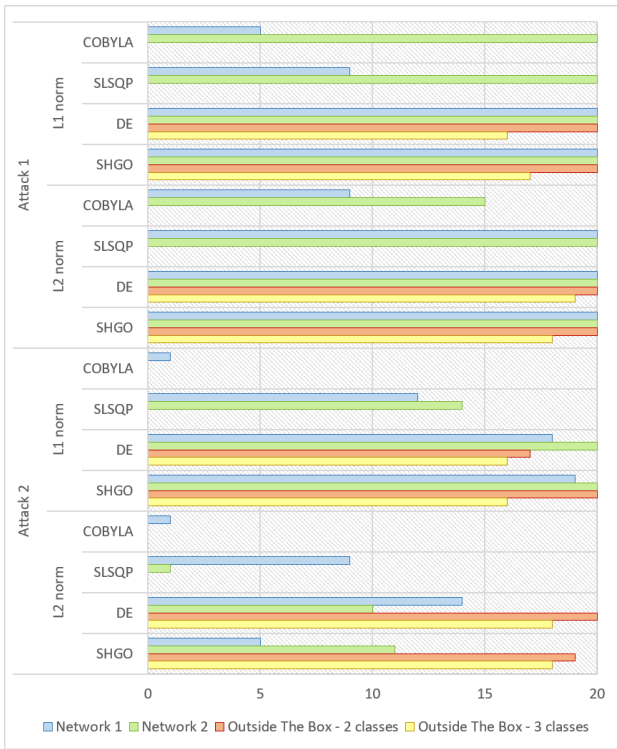


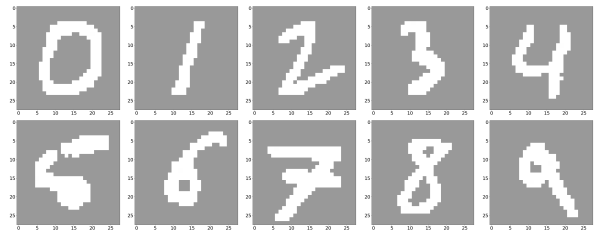
Figure 5: Success rate of different optimisation based methods for the four proposed attacks with different network architectures.

## 4 Experiments

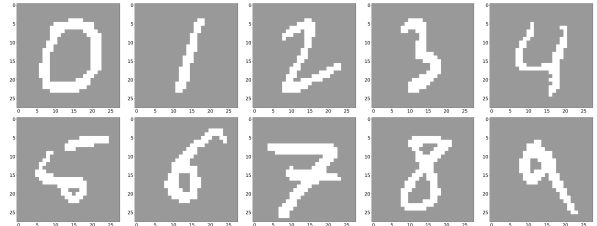
### 4.1 Optimisation solvers experiment

In this experiment, we evaluate the performance of different optimisation solvers in solving the four proposed optimisation problems and successfully generating adversarial samples. We evaluate four solvers from SciPy: COBYLA, SLSQP, SHGO and DE. We experiment with different network architectures, namely two very simple neural networks: a XOR-like circuit and a tangent hyperbolic circuit, and MNIST classifier once trained over two classes, another trained over three classes. The novelty monitor follows "outside the box" paradigm. For each network, we tested over 20 random  $x^0$  samples and counted the times where the optimiser found a solution, i.e. generated an effective attack point  $x$ . Results are shown in Figure 5.

Results show that local optimisation methods such as SLSQP and COBYLA were unsuccessful when applied to the MNIST classifiers. We attribute this to the  $28 \times 28$  dimensionality of the images. Changing many pixels result in a relatively large distance from the original point, which made the search very narrow around the given starting point. Global optimisation methods (SHGO, DE) were much more successful. DE has more successful attempts than SHGO in some cases, but the generated image samples appeared very noisy making the attack easily perceived by a human observer. SHGO is the best method in terms of success rate and preserving the original digit shape. Figure 6 illustrates examples of MNIST images before and after "Attack 1" type (from

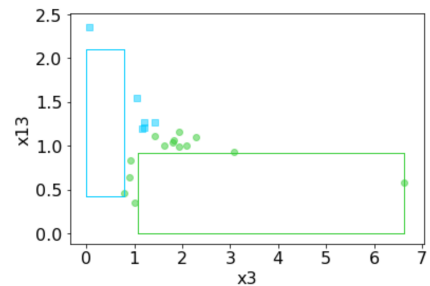


(a) Before SHGO Optimisation Attack. Samples are decided as not novel.

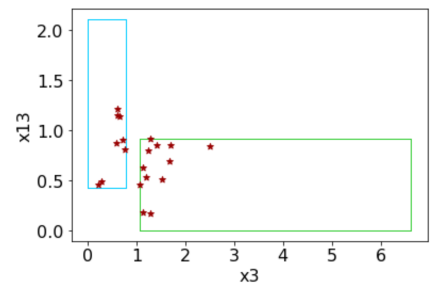


(b) After SHGO Optimisation Attack. Samples are decided as novel.

Figure 6: Adversarial image examples obtained by SHGO method



(a) Attack 1



(b) Attack 2

Figure 7:  $L_1$ -norm optimisation attacks using SHGO method

valid to invalid). The classifier and monitor were trained over the ten classes in this experiment.

Note that even that our objective function is linear, our constraints such as  $\text{monitor}(x) = 0/1$  and  $\text{predict}(x) = \text{predict}(x^0)$  are strongly non-linear, which hinders the task of the used linear solvers. SHGO shined since it is a derivative-free optimiser that is most appropriate for black box functions and leverages input/output pairs. Another comparison factor is the optimiser runtime. We recorded large run-times

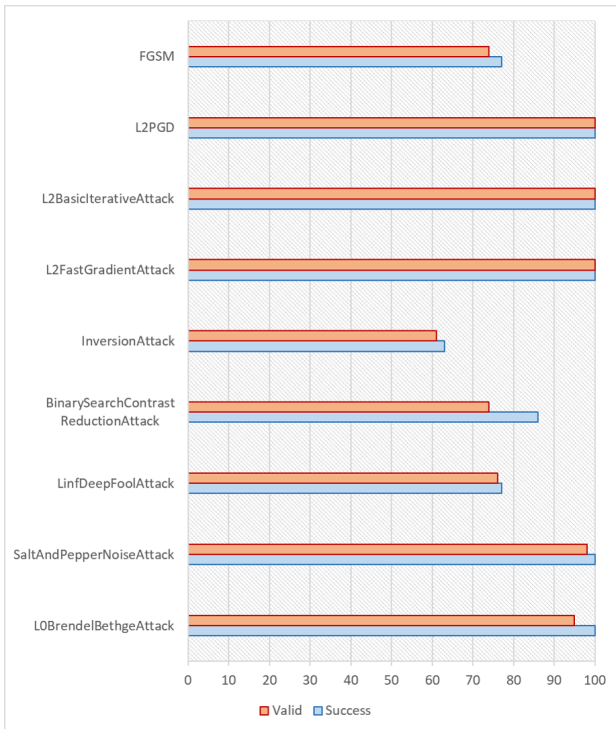


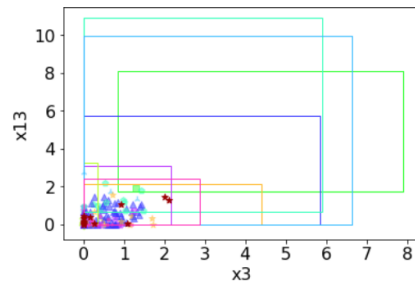
Figure 8: Experiment measuring the number of successful and valid attack attempts using neural network attacks.

(10 – 15 hours) for most optimisers during the experiment using a 16GB ram computer and a processor of 2.60 GHz frequency. However, SHGO converged in matter of few minutes. Figure 7 shows a 2d projection of successful generated examples using SHGO and the  $L_1$ -norm formulation for both types of attacks.

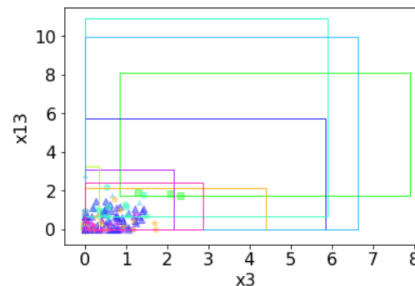
## 4.2 Adversarial attack experiment

In principle, a perfect novelty detector should flag adversarial samples as novel or anomalies rather than validate their wrong predictions. In this experiment, we consider a neural network classifier trained over all the classes of the MNIST dataset. "Outside the box" novelty detection was not designed with the goal of detecting adversarial samples. However, we check whether such samples would be detected as novel or not. Over 100 MNIST samples, figure 8 shows the number of successful attacks (i.e., the prediction was successfully flipped) and the number of successful attacks considered as valid (i.e. the monitor validates the prediction considering the sample as not novel). Results show that most of these samples succeeded into fooling the monitor in addition to fooling the classifier. The initial environment properties of the detector were used, hence a single cluster/box for each class and 0 tolerance. Increasing the number of clusters per class from only one to 2 or 3 clusters per class has a positive impact on invalidating adversarial samples. Conversely, increasing the tolerance factor from zero to 0.1 or 0.25 resulted in accepting more adversarial samples.

Furthermore, the adversarial samples that were success-



(a) Adversarial samples acceptance before optimisation



(b) Adversarial samples acceptance after optimisation

Figure 9: Fooling the classifier and the monitor together.

fully detected as invalid can undergo one of our optimisation attacks to pass the monitoring test. For instance, we ran the  $L_1$  norm "Attack 2" using SHGO on top of FGSM to achieve 100% attack success over 300 MNIST samples. Of course, we had to suppress the  $\text{predict}(\mathbf{x}) \neq \text{predict}(\mathbf{x}^0)$  constraint.

Figure 9 shows a 2d projection of the adversarial samples as per the monitor definition. Before the optimisation attack, there are still adversarial samples marked as  $\star$  points. After the attack, all  $\star$  points disappear.

## 5 Conclusion and Future Work

In this paper, we demonstrated that novelty detection monitors are vulnerable to fooling attacks. We were successfully able to mislead the monitor using multiple methods. We formulated optimisation problems that can be solved efficiently to find attack vectors. We also leveraged adversarial neural networks attacks from the literature to fool the classifier and the monitor at the same time. Adversarial neural network attacks combined with optimisation techniques are shown to be a deadly combo.

The message of the paper is that security by design should be a requirement for new novelty detection systems in deep learning, especially in critical systems. We envision exploring ways to defend novelty detection against adversarial attacks. In future work, we aim at proposing efficient defense mechanisms for novelty detection monitors against both monitor fooling and classifier-monitor fooling attacks.



## References

- [Aggarwal, 2015] Charu C. Aggarwal. *Outlier Analysis*, pages 237–263. Springer International Publishing, Cham, 2015.
- [Ahmed *et al.*, 2016] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.
- [Bodesheim *et al.*, 2015] Paul Bodesheim, Alexander Freytag, Erik Rodner, and Joachim Denzler. Local novelty detection in multi-class recognition problems. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 813–820. IEEE, 2015.
- [Chalapathy *et al.*, 2018] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.
- [Domingues *et al.*, 2018] Rémi Domingues, Pietro Michiardi, Jihane Zouaoui, and Maurizio Filippone. Deep gaussian process autoencoders for novelty detection. *Machine Learning*, 107(8-10):1363–1383, 2018.
- [Endres *et al.*, 2018] Stefan C Endres, Carl Sandrock, and Walter W Focke. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72(2):181–217, 2018.
- [Hawkins *et al.*, 1980] D Hawkins, A Jain, R Dubes, et al. Identification of outliers chapman and hall. *London:[Google Scholar]*, 1980.
- [Henzinger *et al.*, 2020] Thomas A. Henzinger, Anna Lukina, and Christian Schilling. *Outside the Box: Abstraction-Based Monitoring of Neural Networks*, volume 325 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2020.
- [Kraft and others, 1988] Dieter Kraft et al. A software package for sequential quadratic programming. DFVLR Obersfaffehofen, Germany, 1988.
- [Mandelbaum and Weinshall, 2017] Amit Mandelbaum and Daphna Weinshall. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*, 2017.
- [Nassar, 2020] Mohamed Nassar. *Deep Learning Handbook*. Zenodo, 2020. <http://mnassar.github.io/deeplearninghandbook>.
- [Pidhorskyi *et al.*, 2018] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. In *Advances in neural information processing systems*, pages 6822–6833, 2018.
- [Powell, 1994] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.
- [Price, 2013] Kenneth V Price. Differential evolution. In *Handbook of optimization*, pages 187–214. Springer, 2013.
- [Rauber *et al.*, 2020] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020.
- [Roberts *et al.*, 2019] Ethan Roberts, Bruce A Bassett, and Michelle Lochner. Bayesian anomaly detection and classification. *arXiv preprint arXiv:1902.08627*, 2019.
- [Virtanen *et al.*, 2020] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

# Appendix B

## Abbreviations

SLSQP	Sequential Least Squares Programming
SHGO	Simplicial Homology Global Optimization
COBYLA	Constrained Optimization By Linear Approximation
DE	Differential Evolution
FGSM	Fast Gradient Sign Method
L <sub>2</sub> PGD	L <sub>2</sub> ProjectedGradientDescentAttack
KNN-CF	K-nearest neighbors certainty factor
U2R	User to root
R2L	Remote to local

# Bibliography

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.  
<http://www.deeplearningbook.org>.
- [2] M. Nassar, *Deep Learning Handbook*. Zenodo, 2020. <http://mnassar.github.io/deeplearninghandbook>.
- [3] D. Hawkins, A. Jain, R. Dubes, *et al.*, “Identification of outliers chapman and hall,” *London:[Google Scholar]*, 1980.
- [4] C. C. Aggarwal, *Outlier Analysis*, pp. 237–263. Cham: Springer International Publishing, 2015.
- [5] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.
- [6] T. A. Henzinger, A. Lukina, and C. Schilling, *Outside the Box: Abstraction-Based Monitoring of Neural Networks*, vol. 325 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2020.
- [7] A. Mandelbaum and D. Weinshall, “Distance-based confidence score for neural network classifiers,” *arXiv preprint arXiv:1709.09844*, 2017.

- [8] P. Bodesheim, A. Freytag, E. Rodner, and J. Denzler, “Local novelty detection in multi-class recognition problems,” in *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 813–820, IEEE, 2015.
- [9] S. Pidhorskyi, R. Almohsen, and G. Doretto, “Generative probabilistic novelty detection with adversarial autoencoders,” in *Advances in neural information processing systems*, pp. 6822–6833, 2018.
- [10] R. Domingues, P. Michiardi, J. Zouaoui, and M. Filippone, “Deep gaussian process autoencoders for novelty detection,” *Machine Learning*, vol. 107, no. 8-10, pp. 1363–1383, 2018.
- [11] E. Roberts, B. A. Bassett, and M. Lochner, “Bayesian anomaly detection and classification,” *arXiv preprint arXiv:1902.08627*, 2019.
- [12] R. Chalapathy, A. K. Menon, and S. Chawla, “Anomaly detection using one-class neural networks,” *arXiv preprint arXiv:1802.06360*, 2018.
- [13] H. H. Pajouh, G. Dastghaibyfar, and S. Hashemi, “Two-tier network anomaly detection model: a machine learning approach,” *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, 2017.
- [14] I. Rish *et al.*, “An empirical study of the naive bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.
- [15] P. Bodesheim, A. Freytag, E. Rodner, M. Kemmler, and J. Denzler, “Kernel null space methods for novelty detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3374–3381, 2013.

- [16] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [17] D. Kraft *et al.*, “A software package for sequential quadratic programming,” DFVLR Obersfaffeuhofen, Germany, 1988.
- [18] M. J. Powell, “A direct search optimization method that models the objective and constraint functions by linear interpolation,” in *Advances in optimization and numerical analysis*, pp. 51–67, Springer, 1994.
- [19] K. V. Price, “Differential evolution,” in *Handbook of optimization*, pp. 187–214, Springer, 2013.
- [20] S. C. Endres, C. Sandrock, and W. W. Focke, “A simplicial homology algorithm for lipschitz optimisation,” *Journal of Global Optimization*, vol. 72, no. 2, pp. 181–217, 2018.
- [21] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [22] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, “Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax,” *Journal of Open Source Software*, vol. 5, no. 53, p. 2607, 2020.
- [23] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.