Robustness of Autoencoders for Anomaly Detection Under Adversarial Impact

Adam Goodge[1,3] , **Bryan Hooi**[1,2] , **See Kiong Ng**[1,2] and **Wee Siong Ng**[3]

[1]School of Computing, National University of Singapore
[2] Institute of Data Science, National University of Singapore
[3] Institute for Infocomm Research, A*STAR, Singapore
adam.goodge@u.nus.edu, {dcsbhk, seekiong}@nus.edu.sg, wsng@i2r.a-star.edu.sg

Abstract

Detecting anomalies is an important task in a wide variety of applications and domains. Deep learning methods have achieved state-of-the-art performance in anomaly detection in recent years; unsupervised methods being particularly popular. However, deep learning methods can be fragile to small perturbations in the input data. This can be exploited by an adversary to deliberately hinder model performance; an adversarial attack. This phenomena has been widely studied in the context of supervised image classification since its discovery, however such studies for an anomaly detection setting are sorely lacking. Moreover, the plethora of defense mechanisms that have been proposed are often not applicable to unsupervised anomaly detection models. In this work, we study the effect of adversarial attacks on the performance of anomaly-detecting autoencoders using real data from a Cyber physical system (CPS) testbed with intervals of controlled, physical attacks as anomalies. An adversary would attempt to disguise these points as normal through adversarial perturbations. To combat this, we propose the Approximate Projection Autoencoder (APAE), which incorporates two defenses against such attacks into a general autoencoder. One of these involves a novel technique to improve robustness under adversarial impact by optimising latent representations for better reconstruction outputs.

1 Introduction

Anomalies are defined in [Hawkins, 1980] as "*observations which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism*". Anomaly detection is the task of identifying these anomalies in a set of data. It is a crucial task in many applications, from detecting fraudulent credit card usage [Bolton and Hand, 1999] to malignant tumours [Spence *et al.*, 2001], and a plethora of approaches have been developed over time. A common challenge is the absence of labelled data, making supervised classification models ineffective. As such, many of the most successful attempts use a one-class approach; where the model is trained with only normal data and then determines whether test data belongs to the normal class or not. Recently, deep learning methods have given the best performance; particularly popular is to train an autoencoder to reconstruct data of the normal class, using the reconstruction error to determine whether a point is anomalous.

Despite their successes, the highly complex operations of deep learning models can make their outputs fragile to small perturbations in input data. This makes them vulnerable to an adversary who may exploit perturbations that are purposefully designed to greatly hinder model performance. This phenomena, known as an adversarial attack, has drawn a huge amount of attention in related literature since being discovered in [Szegedy *et al.*, 2013] due to the potential risks to model performance and security. Unfortunately, studies have shown that attacks designed for a particular model can generalize to other models with different architectures too [Kurakin *et al.*, 2017]. Many different attacks formulations have been studied in a variety of contexts, a key categorization being between white-box and black-box attacks: whether the adversary uses internal information and computations of the model such as loss gradients, or only inputs and outputs values, respectively. The robustness of these models, how tolerant they are to these input perturbations, is important in a variety of applications. Within the context of anomaly detection, an adversary may wish to perturb input data in such a way that as many anomalies as possible appear normal.

A number of defense mechanisms have been devised against adversarial attacks. Perhaps the most well-known, [Goodfellow *et al.*, 2014] proposes 'adversarial training', which introduces data that has been perturbed with an adversarial attack into the training set along with their correct labels. In doing so, the model is trained to correctly classify attacked examples too. In the case of unsupervised anomaly detection, no labels are supplied to the model and anomalies are only seen during test time, meaning this defense is inapplicable. Therefore, it is necessary to develop new defense mechanisms against adversarial attacks in the unsupervised anomaly detection setting.

With this in mind, the contributions of this work are as follows:

1. We study the vulnerability of anomaly-detecting autoencoders to different types of adversarial attacks.

2. We propose the Approximate Projection Autoencoder (APAE) which includes two developments to improve both model performance and robustness under the impact of adversarial attacks.

In experiments on real data, these techniques led to a median improvement in AUC score of 9% in the presence of adversarial attacks in the range of those tested and 8% in their absence.

2 Background

2.1 Anomaly Detection with Autoencoders

Autoencoders are a very popular approach for anomaly detection. They are neural networks that are trained to reconstruct the input data, with the error between the original and the reconstruction used in the loss function for training:

$$\mathrm{Err}(\mathbf{x}) = ||\mathbf{x} - A(\mathbf{x})||, \tag{1}$$

where $\mathbf{x}$ and $A(\mathbf{x})$ is the input and output of the autoencoder respectively and $||\cdot||$ is typically some type of norm.

There are usually fewer neurons in the hidden layers than the input (and output) layers in order to encourage the model to learn a more compressed representation of the data; a form of dimensionality reduction. As mentioned, it is common to train the autoencoder on only normal data in anomaly detection. In doing so, normal data should be reconstructed with low error during inference whilst anomalies will be reconstructed with higher error as they follow a different distribution. This reconstruction error can be used as an anomaly score; those below or above a given threshold are determined to be normal or anomalous respectively.

2.2 Adversarial Attack

As mentioned, input data can be perturbed as to hinder model performance through adversarial attacks. Formulated in [Kurakin *et al.*, 2017], the 'fast gradient sign method' (FGSM) is particularly prominent. In this attack, inputs are moved in the direction of the gradient of the loss function with respect to these inputs which acts to increase the loss function with respect to the true class and encourage a misclassification. This is extended to the 'basic iterative method', in which this step is performed for multiple iterations as follows:

$$\begin{aligned}&\mathbf{X}_0^{adv} = \mathbf{X},\\ &\mathbf{X}_{N+1}^{adv} = \mathbf{Clip}_{X,\epsilon}\{\mathbf{X}_N^{adv} + \alpha * \mathrm{sign}(\nabla_{\mathbf{X}} J(\mathbf{X}_N^{adv}, y_{true}))\},\end{aligned} \tag{2}$$

where $\mathbf{X}$ is the original input data, J the loss function and y_{true} the true class. This is controlled by a rate α and a clipping function which keeps values in the range $[-\epsilon, \epsilon]$.

3 Related Work

In the supervised setting, anomaly detection is a typical classification problem; the labels corresponding to whether data is normal or anomalous. Well established methods like decision trees and neural networks are applicable in this setting [Zhang *et al.*, 2001], however imbalanced datasets and difficulty obtaining accurate labels make this approach unpopular. As such, unsupervised versions of these types of algorithms have been developed, most notably the One-class SVM [Eskin *et al.*, 2002]. [Goldstein and Uchida, 2016] evaluates a variety of unsupervised anomaly detection methods such as kNN, Local Outlier Factor (LOF) and One-Class SVM.

[Hawkins *et al.*, 2002] first proposes autoencoders for anomaly detection, using a step-function as activation to separate inputs into normal and anomalous clusters. Since then, the reconstruction error as anomaly score has become the norm in more recent studies [Dau *et al.*, 2014]. [Sakurada and Yairi, 2014] show that the non-linearity of autoencoders allows for better detection of anomalies than linear PCA, whilst being computationally cheaper than kernel PCA. Autoencoder variants are also commonly used, including recurrent [Chauhan and Vig, 2015; Malhotra *et al.*, 2015], convolutional [Chen *et al.*, 2018], denoising [Feng and Han, 2015] and variational models [An, 2015]. They have also been used just for feature learning, with a separate prediction network, such as a Gaussian Mixture Model [Bo *et al.*, 2018], using the learnt features to make predictions.

Some studies improve the robustness of anomaly detection models to noisy data, such as Robust SVM [Hu *et al.*, 2003] and Robust PCA. The approach taken in the latter; filtering noise out of input data via matrix decomposition, has also been adopted for autoencoders [Zhou and Paffenroth, 2017]. However, no similar techniques have been developed in the case of adversarial perturbations. [Kloft and Laskov, 2010] studies adversarial poisoning attacks on online centroid anomaly detectors, where the anomaly score is the distance of a point to its nearest centroid. The authors find that, in their intrusion detection scenario, the adversary needs to control 5-20% of incoming traffic in order to subvert an online centroid learner to a target position. [Paudice *et al.*, 2018] uses anomaly detection methods to detect adversarial examples in training sets in order to improve model performance in other tasks as a preparation step, however they do not consider the effect of attacks to the anomaly detection model itself.

4 Methods

4.1 Attacks

The first task is to design adversarial attacks that hinder the performance of autoencoders for anomaly detection. In particular, we consider perturbations to anomalous points in the test set that cause the model to misclassify as many of them as possible as normal points. This is achieved by reducing reconstruction error. Two such attacks are tested: a 'randomized' attack and an 'FGSM' attack.

In the first, a randomly generated vector is added to the original point. If the reconstruction error of this perturbed point is reduced compared to the original, then the perturbation is kept, otherwise it is discarded and another random vector is tested. This is repeated iteratively, so that multiple vectors can be added and the reconstruction error reduces further over iterations.

In the second, we implement an adaptation of the basic iterative method presented in Section 2, which we refer to as

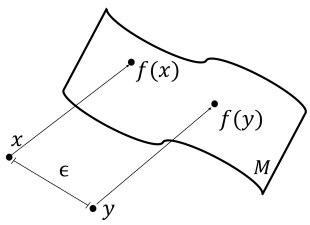

Figure 1: Points $\mathbf{x}$ and $\mathbf{y}$ and their projections $\mathbf{x}'$ and $\mathbf{y}'$ respectively on the image $\mathcal{M}$.

'FGSM' from here for brevity. The adaptation ensures the reconstruction error i.e. the loss function, decreases rather than increases, by moving in the opposite direction of the loss gradient as follows:

$$\begin{aligned} &\mathbf{X}_0^{adv} = \mathbf{X}, \\ &\mathbf{X}_{N+1}^{adv} = \mathbf{X}_N^{adv} - \alpha * \text{sign}(\nabla_{\mathbf{X}} J(\mathbf{X}_N^{adv})). \end{aligned} \tag{3}$$

where $\mathbf{X}$ refers to anomalous points. The clipping function could outweigh the impact of the perturbations and is therefore removed.

4.2 Defenses

We now describe our Approximate Projection Autoencoder (APAE), which combines two defenses: Approximate Projection (AP) and Feature Weighting (FW).

Approximate Projection

Autoencoders reconstruct complex data using lower dimensional representations. Adversarial vulnerability can arise because they can learn highly unstable functions that change rapidly in response to small input perturbations.

In contrast, we consider a class of **projection** functions which are much more robust; moreover, their level of robustness can be characterized theoretically. Formally, let $A : \mathbb{R}^d \to \mathbb{R}^d$ be an autoencoder, and define the **image** $\mathcal{M} = \{A(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^d\}$, i.e. the set of points that the autoencoder can map to. Define the projection $f(\mathbf{x})$ as the closest point in $\mathcal{M}$ to $\mathbf{x}$; i.e. $f(\mathbf{x}) = \arg\min_{\mathbf{x}' \in \mathcal{M}} \|\mathbf{x} - \mathbf{x}'\|$, as visualized in Figure 1. Like in autoencoders, the reconstruction error $\text{Err}(\mathbf{x}) = \|\mathbf{x} - f(\mathbf{x})\|$ is used as an anomaly score; where higher error values indicate more anomalous points. Then the following theorem shows that $f(\mathbf{x})$ is provably robust against adversaries:

Theorem 1. *If an adversary perturbs a data point from* $\mathbf{x}$ *to* $\mathbf{y}$ *such that* $||\mathbf{x} - \mathbf{y}|| \leq \epsilon$*, then we have:*

$$|\text{Err}(\mathbf{y}) - \text{Err}(\mathbf{x})| \leq \epsilon$$

i.e. the adversary can change the reconstruction error of any point $\mathbf{x}$ *by at most* ϵ*.*

Proof. By the triangle inequality we have:

$$\|\mathbf{x} - f(\mathbf{y})\| \leq \|\mathbf{y} - f(\mathbf{y})\| + \epsilon. \tag{4}$$

$f(\mathbf{x})$ is the projection of $\mathbf{x}$ onto $\mathcal{M}$, therefore, it is the closest point to $\mathbf{x}$ on $\mathcal{M}$, so:

$$\|\mathbf{x} - f(\mathbf{x})\| \leq \|\mathbf{x} - f(\mathbf{y})\| \tag{5}$$

Combining this with (4) gives:

$$\|\mathbf{x} - f(\mathbf{x})\| \leq \|\mathbf{y} - f(\mathbf{y})\| + \epsilon; \tag{6}$$

or $\text{Err}(\mathbf{x}) \leq \text{Err}(\mathbf{y}) + \epsilon$. By symmetry, the same result holds when swapping $\mathbf{x}$ and $\mathbf{y}$, completing the proof. □

Theorem 1 shows that in the worst-case scenario, an adversary moving a point by ϵ distance can decrease the reconstruction error under f by at most ϵ. In reality, the projection f is not accessible as the image $\mathcal{M}$ is unknown and highly complex. Instead, after fitting an autoencoder A, we approximate a projection by performing gradient descent on the latent embedding of the data, encoded at the bottleneck layer. Upon convergence, this will lead to a reconstruction that is closer to the optimum; the projection. Gradient descent updates are made via the following formula:

$$\begin{aligned} &\mathbf{Z}_0 = \mathbf{Z}, \\ &\mathbf{Z}_{N+1} = \mathbf{Z}_N - \alpha * \nabla_{\mathbf{Z}} J(\theta, \mathbf{Z}) \end{aligned} \tag{7}$$

where $\mathbf{Z}$ is the original latent embeddings of test set points under A.

Feature Weighting

In the second defense, we acknowledge that different features vary in their discriminative capabilities. In particular, regardless of whether points are normal or anomalous, different features tend to be more accurately reconstructed than others through the autoencoder. It is useful to normalize reconstruction error across the different features in order to account for these differences:

$$\hat{J}_i = \frac{J_i}{\epsilon + \tilde{J}_i} \tag{8}$$

where J_i is the reconstruction error associated with feature i for a single point, $\tilde{J}_i$ is the normalizing factor and ϵ is a small constant. Various formulations of $\tilde{J}_i$ were considered; the median provides robustness against outliers and good empirical performance. The optimal value of ϵ varies between 10^{-4} and 10^{-6} depending on the dataset.

This is a form of normalization which prevents the reconstruction error, and therefore the anomaly score, from being dominated by those features which are reconstructed most poorly regardless of the class. Whilst improving detection performance, this step alone does not necessarily improve robustness against adversarial attacks. However, in combination with the gradient descent defense, the autoencoder would be more accurate as well as more robust in both the presence and absence of adversarial attacks.

5 Experiments

Deep autoencoders are trained using only data of the normal class, using the reconstruction error as the anomaly score. Normal data should be reconstructed with lower error than a chosen threshold, whilst those of the anomalous data should exceed it. We use the k^{th} percentile of reconstruction errors associated with the training set to define this threshold. The value of k is varied within the range of 90 to 100 to explore the difference it makes to the performance.

Dataset	#Features	#Train	#Test	Anomalies
WADI	1220	1048571	172792	5.99%
SWaT	500	496791	449910	11.97%

Table 1: Summary of the two datasets used in experiments.

5.1 Datasets

The Secure Water Treatment (SWaT) is a water treatment testbed resembling those used by Singapore's Public Utility Board. It is an example of a Cyber-Physical System (CPS), integrating digital and physical infrastructure to control and monitor system behaviour. These systems are increasingly used in important sectors, such as utilities or transportation, and are therefore potential targets to attack by malicious actors. The Water Distribution (WADI) is an extension of this setup to include a network of distribution pipelines. There are two weeks worth of data from normal operations, which are used as training data for the respective models. A number of controlled, physical attacks are implemented at different intervals in the following days, which correspond to the anomalies in the test set. More information about the systems and datasets can be found at their websites [iTrust Centre for Research in Cyber Security, 2019].

Table 1 summarises the two datasets, showing the number of features, training and test set sizes as well as proportion of anomalies in the test set. In both cases, a 10 second sliding window (with a stride of one) were concatenated into one feature vector for each data sample - the corresponding label coming from the latest timestamp within the current window. For WADI, 120 sensors are used meaning a total of 1220 features per example. For SWaT, 50 sensors instead make for 500 features.

5.2 Training

Autoencoders with three hidden layers were trained for each dataset. The bottleneck hidden layer had 100 (50) neurons for the WADI (SWaT) dataset, reflecting their difference in input dimensonality. The models are trained using the Adam optimization scheme with learning rate 1×10^{-3} and $(\beta_1, \beta_2) = (0.5, 0.99)$. Early stopping was used once the loss function, Huber loss, is sufficiently small for the validation set, which was 20% of total training data. We use $N = 2000$ iterations and $\alpha = 0.001$ for the gradient descent step (Eq. (7)).

In Table 2, we compare the performance of a standard autoencoder, as well as an autoencoder with our proposed defenses, with various other existing methods in a non-attack setting. These are the following:

APAE: The proposed Approximate Projection Autoencoder with defenses.
AE: Autoencoder as formulated above without defenses.
PCA: Principal Component Analysis finds the set of k orthogonal axes that retain the greatest variance within the data. The anomaly score is the reconstruction error after transforming and inverse-transforming data using these components.
OC-SVM: One-class support vector machine [Chen *et al.*, 2001] adapts the SVM for anomaly detection by maximising the margin between the one-class training data and the origin.

Method	WADI	SWaT
APAE	**0.8711**	**0.9136**
AE	0.805	0.896
PCA	0.816	0.788
OC-SVM	0.730	0.801
DAGMM	0.558	0.683
MAD-GAN	0.568	0.532

Table 2: AUC score for various anomaly detection methods.

DAGMM: Deep Autoencoding Gaussian Mixture Model trains an autoencoder and Gaussian mixture model end-to-end, using energy as the anomaly score as seen in [Bo *et al.*, 2018].
MAD-GAN: A GAN is trained on normal data and the output of the trained discriminator is used to determine anomaly score, as seen in [Liu *et al.*, 2019].

All models were built and implemented using the PyTorch library, except for DAGMM and MAD-GAN where the publicly available codes were used. Even without optimization of hyperparameters, a basic autoencoder is the best of the existing deep methods tested. Even PCA and OC-SVM outperform the other deep methods shown, suggesting that the latter require more meticulous preparation and model calibration to achieve their state-of-the-art performance shown in the original works. We see that in a non-attack setting, the techniques used in the APAE significantly improve the performance of the autoencoder and is easily the most successful model tested. In the following sections, the performance of this model in both attack and non-attack settings is comprehensively studied.

5.3 Attacks

Table 3 show the precision, recall and F1 measures when different reconstruction error percentiles are used as the anomaly score threshold for the two datasets respectively. In the first case, the performance is recorded for the original data without adversarial attack. In this case, we see a general pattern of increasing precision and decreasing recall as the threshold increases. A higher threshold means more anomalies do not exceed it and therefore fewer are correctly detected, however fewer normal points are falsely flagged too. The optimal performance on WADI is achieved with the 99.9^{th} percentile threshold, though it seems a much higher threshold would give better results for SWaT. To avoid this problem, the AUC score is also presented, which eliminates the need for manually-set thresholds. Using this metric, the model for SWaT achieves a score of 0.896 compared to 0.805 for WADI.

In the second case, we apply the randomized attack detailed in Section 4. We conducted 1000 iterations, and therefore 1000 random vectors were generated. These vectors vectors were sampled from a $\mathcal{N}(0, 0.01)$ distribution, as this showed to result in the largest perturbations. This attacks caused a slight reduction in performance, however the most severe setting of the FGSM attack tested is much more impactful, after which the AUC scores reduce by approximately

WADI	Without Attack			Random Attack			FGSM Attack		
Threshold	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
95	0.097	0.814	0.173	0.096	0.803	0.171	0.018	0.138	0.032
99	0.257	0.695	0.375	0.234	0.613	0.339	0.063	0.136	0.086
99.9	0.730	0.540	0.621	0.715	0.500	0.588	0.403	0.135	0.202
100	0.984	0.344	0.510	0.979	0.262	0.414	0.960	0.133	0.233
AUC	0.805			0.764			0.139		
SWaT	Without Attack			Random Attack			FGSM Attack		
Threshold	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
95	0.145	0.974	0.252	0.145	0.974	0.252	0.096	0.607	0.165
99	0.172	0.963	0.292	0.172	0.961	0.292	0.048	0.236	0.080
99.9	0.208	0.897	0.338	0.206	0.886	0.335	0.025	0.089	0.039
100	0.486	0.700	0.573	0.478	0.680	0.562	0.009	0.007	0.008
AUC	0.896			0.890			0.258		

Table 3: Precision, Recall and F1 measures for various anomaly score thresholds for the original test set, the randomly-attacked test set and the FGSM-attacked test set for WADI (top) and SWaT (bottom) datasets.

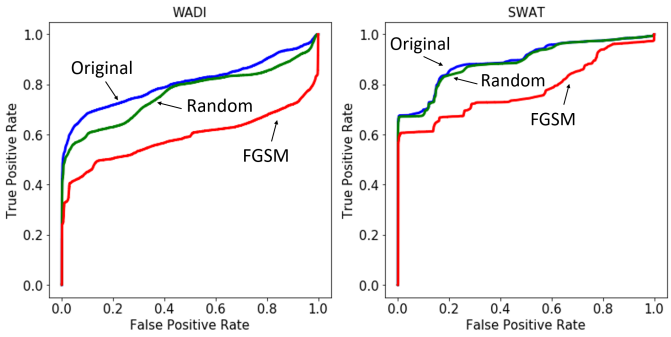

Figure 2: Receiver operating characteristic (ROC) curve for the anomaly-detecting autoencoder on the three test sets used for WADI (left) and SWaT (right).

80% in both datasets. A range of attack severities are tested and discussed in Section 5.4. We see a drastic reduction in the recall, rather than the precision, as the perturbations decrease reconstruction error of anomalies and increase the false negative rate only. Interestingly, the optimal threshold also changes as a result of the FGSM attack; the 100^{th} percentile threshold now giving the best F1 measure for WADI and the 95^{th} for SWaT.

Figure 2 visualises the impact of the randomized attack and a less severe FGSM attack on the ROC curves for both datasets. As the randomized attack had less impact, it will be disregarded from here on and only the FGSM attack will be studied. In the following section, a wide range of implementations of the attack and their quantitative result will be presented, along with the effect of the proposed defenses.

5.4 Defense

Three different attack scenarios are considered. In the first, we consider the case where an adversary can only perturb a limited subset of features of cardinality k, analogous to a subset of sensors in the network. We test a variety of values for k, expressed as a percentage of the total number of features. The attack is iterated until convergence for the k features that had the highest reconstruction error before the first iteration. In the second case, all features can be perturbed; however, the total change, in percentage terms, is limited to a given budget Δ. This percentage is calculated as follows:

$$\text{Perturbation Size} = \frac{\sum_{i=0}^{N} |x_i^{orig} - x_i^{adv}|}{\sum_{i=0}^{N} |x_i^{orig}|},$$

where x_i^{orig}, x_i^{adv} are a point before and after perturbations respectively and N is the total number of features. Summation over all features is taken in order to avoid division by zero. A range of perturbation budgets are tested. The final case is similar to this, however it uses the mean squared error to calculate reconstruction error instead of Huber. As this is differs from the loss function used in model training, this corresponds to a scenario where an adversary does not know how the model was trained and therefore uses the wrong error measure. The three cases are named L_0, L_{huber} and L_2 respectively for convenience.

Table 4 shows the performance of the new method with defenses in these attack scenarios, as well as a no attack case, for the WADI dataset. The best performance is highlighted in bold and underlined and the second best is bold only. The approximate projection (AP) and feature weighting (FW) techniques are tested separately to examine their individual effects, as well as in combination (AP + FW). Furthermore, we distinguish between FW_{Train} and FW_{Test}, which relate to whether the weights are calculated from the feature-wise median reconstruction errors from the training set or the test set. The latter is more applicable to scenarios where data is accessible in batches, whilst the former would be more useful where data is received point-wise, such as in streaming applications. The performance without such defenses is also shown (No Defense).

Firstly without defenses, some general patterns can be observed. As would be expected, performance worsens further with more "powerful" attacks, i.e. with larger k in the case of L_0 and larger Δ in the others. The L_0 attack is noticeably

L_0 attack				
Features ($\approx \%$)	0	1	10	50
No Defense	0.8051	0.8119	0.7933	0.6202
AP	0.8585	0.8589	0.8567	0.7697
FW_{Train}	0.8247	0.8316	0.8094	0.6550
AP + FW_{Train}	**0.8648**	**0.8679**	**0.8666**	**0.7938**
FW_{Test}	0.8501	0.8563	0.8335	0.7241
AP + FW_{Test}	**0.8711**	**0.8770**	**0.8716**	**0.8296**
L_{huber} attack				
Budget Δ (%)	1	5	10	20
No Defense	0.7577	0.5893	0.1884	0.1385
AP	0.8491	0.7345	0.3170	0.1949
FW_{Train}	0.7699	0.6331	0.2056	0.1416
AP + FW_{Train}	**0.8576**	**0.7631**	0.3835	**0.2214**
FW_{Test}	0.7991	0.7120	**0.4250**	0.2040
AP + FW_{Test}	**0.8654**	**0.7943**	**0.5599**	**0.3101**
L_2 attack				
Budget Δ (%)	1	5	10	20
No Defense	0.7665	0.6873	0.5591	0.0004
AP	0.8519	0.8075	0.7094	0.0550
FW_{Train}	0.7795	0.7118	0.6061	0.0006
AP + FW_{Train}	**0.8597**	**0.8225**	**0.7420**	**0.0755**
FW_{Test}	0.8084	0.7550	0.6968	0.0409
AP + FW_{Test}	**0.8660**	**0.8405**	**0.7800**	**0.1586**

Table 4: WADI: AUC for the three different types of attack with and without defenses.

L_0 attack				
Features ($\approx \%$)	0	1	10	50
No Defense	0.8955	0.8438	0.8438	0.7750
AP	0.8935	0.8412	0.8410	0.7750
FW_{Train}	0.8982	0.8481	0.8478	0.7779
AP + FW_{Train}	0.8959	0.8472	0.8470	0.7839
FW_{Test}	**0.9126**	**0.8879**	**0.8877**	**0.8368**
AP + FW_{Test}	**0.9136**	**0.8852**	**0.8851**	**0.8292**
L_{huber} attack				
Budget Δ (%)	10	20	50	80
No Defense	0.8766	0.8588	0.7788	0.2577
AP	0.8776	0.8624	0.7913	0.2541
FW_{Train}	0.8786	0.8628	0.7995	0.3544
AP + FW_{Train}	0.8792	0.8670	0.8119	0.3605
FW_{Test}	**0.8986**	**0.8895**	**0.8375**	**0.4899**
AP + FW_{Test}	**0.8996**	**0.8906**	**0.8414**	**0.4948**
L_2 attack				
Budget Δ (%)	10	20	50	80
No Defense	0.8772	0.8590	0.7853	0.1973
AP	0.8786	0.8596	0.7938	0.1901
FW_{Train}	0.8792	0.8614	0.8056	0.2863
AP + FW_{Train}	0.8780	0.8632	0.8129	0.2907
FW_{Test}	**0.8987**	**0.8890**	**0.8406**	**0.4249**
AP + FW_{Test}	**0.8999**	**0.8907**	**0.8443**	**0.4202**

Table 5: SWaT: AUC for the three different types of attack with and without defenses.

less impactful than the others, despite having an unrestricted perturbation budget. Surprisingly, the AUC score actually increases from perturbing 0% (i.e. no attack) to 1% of features. For SWaT, perturbing 1% or 10% of features made negligible difference.

L_{huber} attacks caused greater deterioration in performance than L_2 attacks for smaller Δ in both datasets, however the inverse is true for the highest budgets tested. It was noticed that in both datasets, some anomalous intervals had drastically larger reconstruction errors than others, suggesting a significantly larger perturbation would be required to reduce it to a normal level. The L_2 loss more heavily penalises these extreme values, therefore it is more likely to reduce these extreme errors sufficiently when given a large enough budget compared with L_{huber}.

Moving onto the defenses, both gradient descent and feature weighting improve the performance across the range of attacks. Considered in isolation, AP fares better than both forms of FW, and FW_{Test} is better than FW_{Train} overall. The best performance is found when combining AP + FW_{Test}.

Table 5 shows the same results for SWaT. Generally, the same pattern applies here as to WADI. However, it was noticed that much larger perturbation budgets were needed for a significant impact to performance. Furthermore, the AP defense is less successful on this dataset; performing worse than no defense in the L_0 attacks. In these cases, the best performance is found using FW_{Test} without AP.

6 Conclusion

We have examined the problem of adversarial impact to deep autoencoders for anomaly detection. Deep autoencoders are trained to learn patterns from a set of data of only the normal class. The reconstruction error associated with a point in the test set is used to determine whether it is normal or anomalous. We have considered the context in which an adversary perturbs this data in order to have as many anomalies go undetected as possible. We have shown that the model is indeed vulnerable to this kind of attack, especially to a gradient-based attack like the basic iterative method. We have also shown that through analysing the worst-case scenario of an attack, the change in reconstruction error associated with an attacked point is bounded by the size of the perturbation if we assume the autoencoder has learnt an optimal mapping, i.e. projection. We propose the Approximate Projection Autoencoder, which involves implementing gradient descent on latent embeddings as a way to optimise reconstructions. We also account for the natural variability in reconstruction errors between different features through a feature-weighting normalization step. Experiments with real data show that combining both defenses almost always improves anomaly detection performance by the largest amount, regardless of the presence or severity of an attack.

Acknowledgements

This work was supported in part by NUS ODPRT Grant R252-000-A81-133.

References

[An, 2015] Jinwon An. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. In *SNU Data Mining Center 2015-2 Special Lecture on IE*, 2015.

[Bo *et al.*, 2018] Zong Bo, Qi Song, and H Chen. Deep autoencoding gaussian mixture model for unsupervisedanomaly detection. 2018.

[Bolton and Hand, 1999] Richard Bolton and David Hand. Unsupervised profiling methods for fraud detection. In *Proceedings of the Conference on Credit Scoring and Credit Control VII*, 1999.

[Chauhan and Vig, 2015] Sucheta Chauhan and Lovekesh Vig. Anomaly detection in ECG time signals via deep long short-term memory networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–7, 2015.

[Chen *et al.*, 2001] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class SVM for learning in image retrieval. In *International Conference on Image Processing*, pages 34–37. Citeseer, 2001.

[Chen *et al.*, 2018] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*, pages 1–5, 2018.

[Dau *et al.*, 2014] Hoang Anh Dau, Vic Ciesielski, and Andy Song. Anomaly detection using replicator neural networks trained on examples of one class. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 311–322. Springer, 2014.

[Eskin *et al.*, 2002] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[Feng and Han, 2015] Wenhui Feng and Chongzhao Han. A Novel Approach for Trajectory Feature Representation and Anomalous Trajectory Detection. *International Conference on Information Fusion*, pages 1093–1099, 2015.

[Goldstein and Uchida, 2016] Markus Goldstein and Seiichi Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE*, 11(4):e0152173, apr 2016.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Hawkins *et al.*, 2002] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. *DaWaK2002. LNCS*, 2454:170–180, 2002.

[Hawkins, 1980] Douglas Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[Hu *et al.*, 2003] Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, pages 282–289, 2003.

[iTrust Centre for Research in Cyber Security, 2019] iTrust Centre for Research in Cyber Security. Datasets, 2019.

[Kloft and Laskov, 2010] Marius Kloft and Pavel Laskov. Online Anomaly Detection under Adversarial Impact. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 405–412, 2010.

[Kurakin *et al.*, 2017] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR 2017*, 2017.

[Liu *et al.*, 2019] Lingbo Liu, Zhilin Qiu, Guanbin Li, Qing Wang, Wanli Ouyang, and Liang Lin. Contextualized Spatial-Temporal Network for Taxi Origin-Destination Demand Prediction. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–13, 2019.

[Malhotra *et al.*, 2015] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *ESANN 2015 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 89–94. Presses universitaires de Louvain, 2015.

[Paudice *et al.*, 2018] Andrea Paudice, Luis Munoz-Gonzalez, Andras Gyorgy, and Emil Lupu. Detection of Adversarial Training Examples in Poisoning Attacks through the Anomaly Detection. 2018.

[Sakurada and Yairi, 2014] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[Spence *et al.*, 2001] Clay Spence, Lucas Parra, and Paul Sajda. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, 2001.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[Zhang *et al.*, 2001] Zheng Zhang, Jun Li, C.N. Manikopoulos, Jay Jorgenson, and Jose Ucles. HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proc. IEEE Workshop on Information Assurance and Security*, pages 85–90, 2001.

[Zhou and Paffenroth, 2017] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674. ACM, 2017.