Robust Methods for Anomaly Detection

with Applications to Cyber Data

by

Chong Zhou

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Data Science

April 19, 2019

APPROVED:

Professor Randy C. Paffenroth Worcester Polytechnic Institute Advisor	Professor Xiangnan Kong Worcester Polytechnic Institute Committee Member
Professor Yanhua Li Worcester Polytechnic Institute Committee Member	Dr. Partha Pal Raytheon BBN Technology External Committee Member
Professor Elke A. Rundensteiner Worcester Polytechnic Institute Program Director	

Abstract

In many real-world problems, large outliers and pervasive noise are commonplace, and *one may not have access to clean training data*. Accordingly, anomaly detection methods are useful to detect and remove anomalies for further analysis. Robust Principal Component Analysis (Robust PCA) is an example of such a method that splits data into a sparse anomaly part and the remaining part which can be projected on a *linear* low-dimensional manifold.

Our work consists of both methodology development and real-world applications. We generalize Robust PCA from discovering linear manifolds to non-linear relationships in the data. In the recent literature, deep autoencoders and other deep neural networks have demonstrated their effectiveness in exploring *non-linear* features across many problem domains. Our extension combines deep autoencoders and Robust PCA, which not only maintain a deep autoencoders' ability to discover non-linear features, but can also eliminate noise. We present generalizations of our results to grouped sparsity norms which distinguish anomalies from structured corruptions, such as a collection of instances having more corruptions than their fellows. Leveraging grouped norms allows our method to detect row-wise outliers. Both denoising and outlier detecting increase the robustness of standard deep autoencoders, and we named our novel method a "Robust Deep Autoencoder (RDA)". This work has been published as a full paper on the research track of the KDD'17 conference. Further, we propose a model consisting of a hierarchical collection of RDAs which maintains the spirit of stacked denoising autoencoders and hierarchical neural networks. By allowing any advanced autoencoder, such as a sparse autoencoder or a variational autoencoder, to name but a few, to replace the standard autoencoders used previously in the RDA framework, we demonstrate that the RDA framework can be expanded to a wide range

of deep models. These models include, but are not limited to, grouped norm regularized sparse autoencoders and variational generative models.

On the aspect of practical employments, we present real-world applications of Robust PCA and RDA to the cyber security domain to analyze dimensionality of data and find anomalies. We detect anomalies in data arising from high fidelity simulation networks and both Robust PCA and RDA provide effective features capable of identifying different Distributed Denial of Service (DDoS) attacks. Finally, we provide a procedure for modifying Robust PCA to adapt to dynamic streaming data. Our contribution has been built into the Adaptive Resource Management Enabling Deception (ARMED) system which aims to enable better detection and mitigation of DDoS attacks.

Acknowledgments

I sincerely thank my adviser, Prof. Randy C. Paffenroth, for multiple fruitful and inspiring discussions, his wise comments and advise on numerous preliminary versions of my publications. I thank him for pushing me and encouraging me to complete my Ph.D. thesis. He played a very important role in every step of my Ph.D. studies from my first class at WPI, through every milestone, publication, etc. He was both supportive and encouraging while supervising my research from a vague idea to a polished publication in a top-level venue. Without his help, I would not be where I am now in my professional and personal development.

I would like to thank my committee members, Prof. Xiangnan Kong, Prof. Yanhua Li, and Dr. Patha Pal, for careful reading of my publications and this manuscript, and improving them by their numerous suggestions.

I would also like to thank Prof. Elke A. Rundensteiner and Data Science program of Worcester Polytechnic Institute, for supporting me as a teaching assistant during the first two years of my studies and researches. I thank Dr. Partha Pal, Raytheon Company, and DARPA (contract No. HROOll-16-C-0058), for funding me as a research assistant in the second two years of my research.

I would like to thank my collaborators, Haitao Liu, Hongzhu Cui, Cassidy Litch, Cody Doucette, Huimin Ren, Yun Yue, and Xiao Qin, for their brilliant ideas, critical comments and productive collaborations. They helped me with elaborating and polishing explanation of my publications. It was a pleasure to work with them, and I am looking forward to our future joint publications.

I would also like to thank members of our research group, especially Fan Yang, Wenjing Li, Yingnan Liu, Matthew L. Weiss, Nitish Bahadur, Kelum Gajamannage, F. Patricia Medina, and Xiaozhou Zou, for listening to my preliminary and final

talks and enriching them with bright ideas and critical comments.

I am endlessly grateful to my family and friends, Pu Zhou, Jiangrong Wu, Xianglin Wu, Hanqin Zhou, Yuehua Zhang, Nan Li, Han Jiang, Hongnan Li, Guojun Wu, Menghai Pan, Hang Cai, and Xinyuan Sun, for their everlasting support during my Ph.D.. This dissertation is devoted to them.

Last but not least, I thank the administrative assistant of Data Science Program, Mary Racicot, for her kind help and administrative work.

Publications Contributing to this Dissertation

Paper Published

Chong Zhou and Randy C. Paffenroth. "Anomaly detection with robust deep autoencoders." *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 2017.

Cui, Hongzhu, **Chong Zhou**, Xinyu Dai, Yuting Liang, Randy Paffenroth, and Dmitry Korkin. "Boosting gene expression clustering with system-wide biological information: a robust autoencoder approach." *International Journal of Computational Biology and Drug Design.*

Paper in Submission

Chong Zhou and Randy C. Paffenroth. "Hierarchical and Robust Autoencoders." In submission to IEEE Transactions on Pattern Analysis and Machine Intelligence (2018).

Chong Zhou, Cassidy Litch, Randy C. Paffenroth, Partha Pal, Nathanial Soule and Regan Broderick-Sander. "Deep and Shallow Robust Methods for Detecting Distributed Denial of Service (DDoS) Attacks." In submission to ACM Digital Threats: Research and Practice (2019).

Cody Doucette, Regan Broderick-Sander, Benjamin Toll, Aaron Helsinger, Nathaniel Soule, Partha Pal, **Chong Zhou**, and Randy Paffenroth. "Anomaly Detection in ARMED using Robust Principal Component Analysis" (2019).

Huiming Ren, Yue Yun, **Chong Zhou**, Randy Paffenroth, and Yanhua Li. "Robust Variational Autoencoders." In submission to ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019).

Liu, Haitao, Randy C. Paffenroth, Jian Zou, and **Chong Zhou**. "Anomaly Detection via Graphical Lasso." In submission to Transactions on Signal Processing (2018).

Executive Summary

We foreshadow our work with a high-level summary and key results of this dissertation. In this study, we are targeting the case where one wishes to train algorithms when one only has access to noisy data. In particular, the noise and outliers are commonplace which significantly reduce the quality of data, while cleaned, noise-free data are expensive and require extra manual processing. Herein, we address this problem with a solution which is structured in three parts, a set of novel models, an applicable training algorithm, and their applications to cyber security. An outline of my work can be found in Figure 1.

The left side of Figure 1 lists some important works in the *literature* including Robust PCA, Deep Autoencoders, Sparse Autoencoders, and Variational Autoencoders that inspire us to build our contribution. Some key ideas of these work are introduced in Section 3.

My first innovative work, shown in the green boxes, starts with a new combination of Robust PCA and deep autoencoder which results in a novel model that allows denoising *without any noisy-free data*, and we name our model a "Robust Deep Autoencoders(RDA)". We develop a novel training algorithm that achieves fast and stable performance. As a consequence of group norms such as the $\ell_{2,1}$ norm, we extend RDA to a novel outlier detection method. This work has been published in KDD'2017 and will be detailed in Section 5 and 11.

Our next stage of work mainly consists of forming a hierarchical model based on RDA and extending RDA with sparse models. Each layer in such a hierarchical model continuously improves performances of denoising or outlier detection and the final output will be as clean as possible. This part of our work will be detailed in Section 7 and 6 and resulted in a paper which was submitted to IEEE Transaction

on Pattern Analysis and Machine Intelligence.

Further work also demonstrates our RDA is extendable to generative models. Collaborating with Huimin Ren and Yue Yun, we build new contribution that combines RDA and variational autoencoders which allows one to *generate new and clean samples from only noisy inputs.* This work formed a paper which was submitted to KDD'2019 and will be introduced in Section 8.

The blue boxes in Figure 1 show the application aspect of my research in which we apply RPCA and RDA for analyzing DDoS attacks. In particular, we find that both dimensionality and anomalies provided by RPCA can be sensitive barometers to cyber attacks. This RPCA-based analysis is a key component of the ARMED (Adaptive Resource Management Enabling Deception) technology suite developed under the DARPA XD3 program. *ARMED is currently being evaluated for transition to defend a tactical situational awareness server that is widely used by various civilian and military agencies.*

Collaborating with Cassidy Litch, we showed that features generated from RDA can significantly reduce false positive rates of predictions on cyber data. The results are demonstrated in Section 15 and 16. We submitted this work to ACM Digital Threats: Research and Practice.

The red boxes in Figure 1 shows some additional works that either unpublished or I am not the primary author. In particular, we modify the RPCA and allow it detecting anomalies in the streaming environments, which has been finished and it is in preparation to submission.

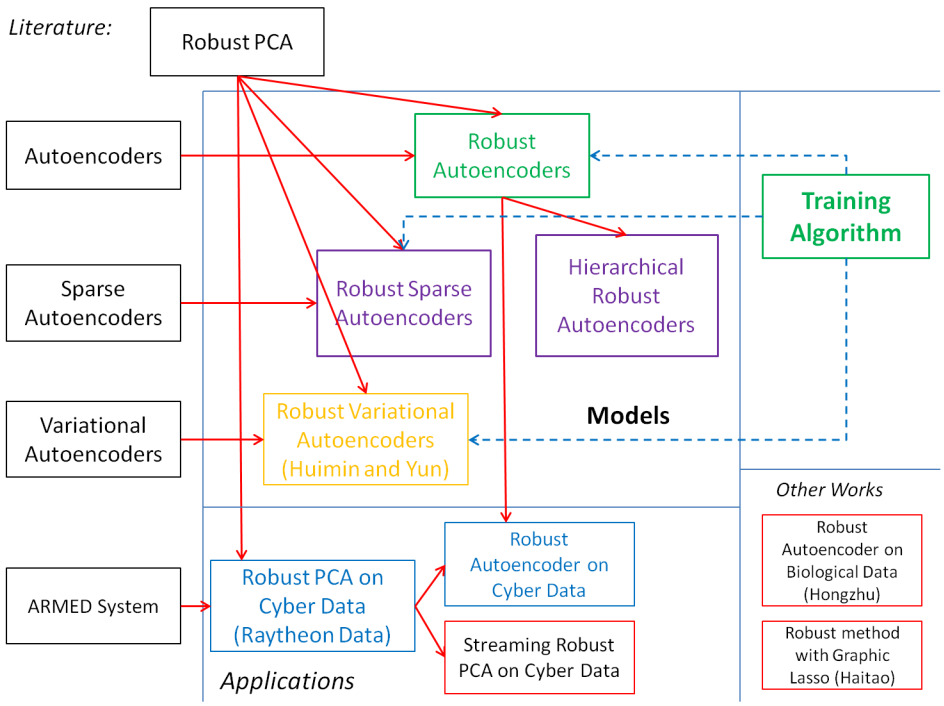

Figure 1: This figure shows the road map of my research. The left side lists some important results in the literature including Robust PCA, Deep Autoencoders, Sparse Autoencoders, and Variational Autoencoders that inspire us to build our contribution. My first innovative work, shown in green boxes, includes the Robust Deep Autoencoder (RDA), which is a new combination of Robust PCA and deep autoencoder, and an efficient training algorithm which was accepted to KDD'2017. The purple boxes shows that our next stage work consists of forming a hierarchical model based on RDA and extending RDA with sparse models with has been submitted to IEEE: Transaction on Pattern Analysis and Machine Intelligence. The orange box show a collaborative work with Huimin Ren and Yun Yue, which demonstrate a variational autoencoder and has submitted to KDD'2019. The blue boxes are an application project that we apply Robust PCA and RDA for analyzing DDoS attacks. The other boxes in red are either unpublished or I am not the primary author.

Contents

List of Figures

List of Tables

1 Introduction

Deep learning is part of a broad family of methods for representation learning [34], and it has been quite successful in pushing forward the state-of-the-art in multiple areas [24, 25, 34, 60]. Unfortunately, outliers and noise may reduce the quality of representations discovered by deep autoencoders [41, 42] and classic denoising autoencoders require access to anomaly-free data which may not be available for many anomaly detection problems. Herein, we provide our solution that develops novel robust models, offers training algorithm, and excises on cyber data.

First, to detect anomalies and provide anomaly-free data for further analyzing, we develop a family of robust models that not only detect anomalies but also discover high quality *non-linear* features. Robust Principal Component Analysis (Robust PCA) is classically used to construct low-dimensional *linear* features by filtering out outlying measurements [48]. Robust PCA is a generalization of Principal Component Analysis (PCA) that attempts to reduce the sensitivity of PCA to anomalies [48]. In particular, Robust PCA allows for the careful teasing apart of sparse anomalies so that the remaining low-dimensional approximation is faithful to the noise-free low-dimensional subspace describing the bulk of the raw data [48].

Combining Robust PCA and deep autoencoders, we proposed "Robust Deep Autoencoders" (RDAs) [69] that extend normal autoencoders and filter out anomalies. Such methods isolate noise and outliers in the input, and the autoencoder is trained iteratively with this isolation. These methods provide a representation at the hidden layers which is more faithful to the true representation of the noise-free data [69].

Stacked autoencoders construct representations based on multiple non-linear

autoencoders [60]. We proposed a hierarchical model which allows the stacking of RDAs, where each layer of the stacked RDA builds upon the representation from the previous layer. In effect, each layer is given the cleaned data from the previous layer, it then attempts to detect any additional anomalies missed by the previous layer. We demonstrate how this layer-wise isolation of RDAs enhances the overall effectiveness of the anomaly detection system.

We also proposed a novel sparse autoencoder model which leverages an $\ell_{2,1}$ regularization term on the hidden layers and gives rise to more concise and informative hidden features. In particular, this model provides a sparse autoencoder which is more faithful to classic dimension reduction in that it identifies a consistent set of important features across all training data. We demonstrate how this $\ell_{2,1}$ sparse autoencoder can also be used in the RDA framework discussed above, and we name this new combination a "Robust Sparse Autoencoder (RSA)".

However, these robust approaches give rise to new training challenges that isolating anomalies cannot be efficiently computed by back-propagation [56] which is a common training method for deep models. As a consequence, we derive a training algorithm for the proposed models by combining ideas from proximal methods [8], back-propagation [56], and the Alternating Direction of Method of Multipliers (ADMM) [7].

Finally, we demonstrate how Robust PCA and RDA can be used to analyze the dimension and detect anomalies in data arising from Raytheon BBN's high-fidelity simulation infrastructure and make use of RPCA derived features to identify different DDoS attacks. We also present our modification of RPCA that allows RPCA to process data in streaming network environments.

Part I

Robust Deep Neural Network Models

2 Overview

Recent literature suggests that noise and outliers reduce the efficiency of deep neural networks [69]. Herein, we address this problem in a way which is inspired by Robust Principal Component Analysis (RPCA). In particular, the $X = L+S$ framework of RPCA, where X is the original data, S is a presentation of the anomalies, and L is the remaining noise-free part, allows us to extend multiple deep models with an ability to resist anomalies. In Section 3, we begin with a concise introduction of Robust Principal Component Analysis which is the "prototype" model of the $X = L + S$ framework and also include key ideas of some popular deep models including Denoising Autoencoders [59, 60] , Sparse Autoencoders [46], Stacked Denoising Autoencoders [60] and Variational Autoencoders [31, 20]. From Section 5 to Section 8, we sequentially present our proposed methodologies that deploy $X = L + S$ framework onto standard deep models, and argue the performances about their resistance to anomalies with experimental results.

3 Background

In this section, we introduce various techniques that form the foundation of our work. In particular, there is a vast literature on RPCA and various kinds of autoencoders, and we have been inspired by those methods. In particular, here we will describe RPCA [48], Deep Autoencoders [25], Sparse Autoencoder [46], and

Variational Autoencoders [31].

3.1 Robust Principal Component Analysis

Robust Principal Component Analysis (RPCA) [15, 48] is a generalization of Principal Component Analysis (PCA) [23] that attempts to reduce the sensitivity of PCA to outliers. Herein, we outline some of the key ideas from RPCA which are classically used to construct low-dimensional *linear* features by filtering out outlying measurements using a convex relaxation of the rank operator [15, 48]. In particular, RPCA allows for the careful teasing apart of sparse anomalies so that the remaining low-dimensional approximation is faithful to the noise-free low-dimensional subspace describing normal data [15, 48].

RPCA splits a data matrix X into a low-rank matrix L and a sparse matrix S such that

$$X = L + S. \tag{1}$$

The matrix L contains a low-dimensional representation of X [15, 48] and the matrix S consists of element-wise anomalies, which cannot be efficiently captured by the low-dimensional representation L.

Naively, this matrix decomposition can be computed by way of the following optimization problem [15]

$$\begin{aligned} &\min_{L,S} \rho(L) + \lambda ||S||_0 \\ &\text{s.t. } \|X - L - S\|_F^2 = 0, \end{aligned} \tag{2}$$

where $\rho(L)$ is the rank of L, $||S||_0$ is the number of non-zero entries in S, and $\|\cdot\|_F$

is the Frobenius norm. However, the non-convex optimization (2) is NP-hard [15] and only computationally tractable for small matrices X. However, there is a large literature[15, 48] on convex relaxations of this class of problems, such as

$$\begin{aligned} &\min_{L,S} \|L\|_* + \lambda \|S\|_1 \\ &\text{s.t. } \|X - L - S\|_F^2 = 0, \end{aligned} \tag{3}$$

where the $\|\cdot\|_*$ is the nuclear norm (i.e., the sum of the singular values of the matrix) and $\|\cdot\|_1$ is the one norm (i.e., the sum of the absolute values of the entries).

3.2 Deep Autoencoders

Deep models, especially deep autoencoders, have demonstrated their effectiveness in discovering non-linear features across many problem domains. We begin the derivation of our methods by providing a brief introduction to non-linear feature representation by way of deep autoencoders.

A deep autoencoder is a feed-forward multi-layer neural network in which the desired output is the input itself. Upon first glance, this process may seem trivial since the identity mapping would also yield exactly the same input data. However, autoencoders become non-trivial when the identity map is disallowed either by way of some type of regularization or, more importantly for the current derivation, by having hidden layers which are *low-dimensional*, non-linear representations of the input data.

In particular, autoencoders learn a map from the input to itself through a pair of encoding and decoding phases

$$\overline{X} = D(E(X)), \tag{4}$$

where X is the input data, E is an encoding map from the input data to the hidden layer, D is a decoding map from the hidden layer to the output layer, and $\overline{X}$ is the recovered version of the input data. The idea is to train E and D to minimize the difference between X and $\overline{X}$.

In particular, an autoencoder can be viewed as a solution to the following optimization problems:

$$\min_{D,E} \|X - D(E(X))\|, \tag{5}$$

where $\|\cdot\|$ is commonly chosen to be the ℓ_2-norm.

Usually, an autoencoder with more than one hidden layer is called a deep autoencoder [34] and each additional hidden layer requires an additional pair of encoders $E(\cdot)$ and decoders $D(\cdot)$. By allowing many layers of encoders and decoders, a deep autoencoder can effectively approximate complicated distributions over the input X. In the sequel, our focus will be on deep autoencoders with all autoencoders assumed to be deep.

3.3 Sparse Autoencoders

Sparse autoencoders pursue sparse representations by imposing regularization terms on hidden layers [38]. A typical sparse autoencoder is defined as

$$\min_{D,E} \|X - D(E(X))\| + \beta\phi(E(X)), \tag{6}$$

where the notation is the same as defined in equation (5) except $\phi(E(X))$ is a sparsity regularization term on the encoded hidden layers and β controls the influence of such a regularization. It is important to note that there are multiple choices for

$\phi(\cdot)$. For example, we could use the ℓ_1 norm [38], the KL-divergence [46], or, as we proposed the $\ell_{2,1}$ norm. β controls the weight of sparsity regularization. Large values of β will encourage less active values in hidden layers and produce low-dimensional representations. On the other hand, small values of β will encourage more active values in hidden layers and allow the mapping to be high-dimensional [46].

Typical sparse autoencoders use the ℓ_1 norm [38] or the KL-divergence [46] as sparsity regularizations on their hidden layers. These regularizations force each instance to use a few features on each hidden layer to recover features of the input layer [46]. We propose that sparse autoencoder with $\ell_{2,1}$ regularization will eliminate features which are less informative since the $\ell_{2,1}$ norm *groups each feature and introduces sparsity between features*. Both ℓ_1 and KL-divergence regularized sparse autoencoders have been successful in discovering informative features in the literature [38, 46]. However, in outlier detection tasks, we assume the normal instances are similar, and outliers are strange and share less information with the normal instances. Thus, the normal instances can be represented by using a small number of similar features, while outlier cannot.

With the definition of the $\ell_{2,1}$ norm (13) in mind, our $\ell_{2,1}$ regularized sparse autoencoders can be defined as

$$\min_{D,E} \|X - D(E(X))\| + \beta \|E(X)\|_{2,1}, \tag{7}$$

where the terms are defined in equation (6) except the $\phi(E(X))$ has been specified as $||E(X)||_{2,1}$.

To discover the non-linear and *low-dimensional* hidden layer, we introduce an $\ell_{2,1}$ norm as a regularization term on hidden layers of sparse autoencoders and

we name our proposed model "$\ell_{2,1}$ Sparse Autoencoder". The $\ell_{2,1}$ norm offers sparsity of hidden layer features that only allows a small number of features for every instance to choose from.

3.4 Denoising Autoencoders

Deep Autoencoders are commonly used for non-linear feature selection and extraction. However, when there are more nodes in the hidden layer than there are inputs, deep autoencoders suffer from the difficulties in learning useful intermediate representations by an initial unsupervised learning step [59].

Based on the idea of making the learned representations robust to partial corruption of the input pattern, Vincent etc. introduced Denoising Autoencoders [59]

$$\min_{D,E} \|X - D(E(\tilde{X}))\|, \tag{8}$$

where the terms are defined in equation (5) except the input X has been modified as its corrupted version $\tilde{X}$.

The objective of Denoising Autoencoder however is fundamentally different from that of developing a image denoising algorithm. In particular, Denoising Autoencoder obtain desired robustness by explicitly introduce corrupting noise as a novel criterion guiding the learning representations. Thus such a corruption+denoising procedure is applied not only on the input, but also recursively to intermediate representations.

3.5 Variational Autoencoders

Generative models are a broad area of machine learning which deals with generate new samples from current data [20]. One major challenge of such models is that the data X is in some potentially high-dimensional space which results in its distributions, $p(X)$, is intractable to compute with a limited number of observations. A Variational Autoencoder (VAE) [31, 54] solves this problem by assuming all the observed instances X are generated from a tractable latent variable z. In particular, $p(z)$ is the prior distribution of the latent variable, $q(z|x)$ is the approximate inference mapping and $p(x|z)$ is the generative mapping. Traditionally, $p(z)$ is often assumed to be a simple distribution such as the Gaussian distribution that allows easy resampling. The VAE parameterizes $q(z|x)$ and $p(x|z)$ by neural networks as

$$\begin{aligned} q(z|x) &= E_{\theta_1}(x) \\ p(x|z) &= D_{\theta_2}(z), \end{aligned} \tag{9}$$

where E_{θ_1} and D_{θ_2} are two neural networks parameterized with θ_1 and θ_2 respectively. In analogy to autoencoders, E_{θ_1} is called the encoder and D_{θ_2} is called the decoder. With ideas from [31, 54], the commonly used optimization function for VAE training is:

$$\min_{\theta_1,\theta_2} \|X - D_{\theta_2}(E_{\theta_1}(X))\| + KL(E_{\theta_1}(X) \mid \mathcal{N}(0,1)), \tag{10}$$

where KL represents Kullback-Liebler divergence (KL divergence) and the first term, $\|X - D_{\theta_2}(E_{\theta_1}(X))\|$ represents the reconstruction errors.

4 Related Work

4.1 Robust Deep Autoencoders

In many real-world problems, large outliers and pervasive noise are commonplace in the data. There is a large extension of literature which attempts to address this challenge and two promising approaches are *denoising autoencoders* and *maximum correntropy autoencoders*. Denoising autoencoders [24, 44, 55, 62, 68], require access to a source of clean, noise-free data for training, and such data is not always readily available in real-world problems [60]. On the other hand, maximum correntropy autoencoders replace the reconstruction cost with a noise-resistant criteria *correntropy* [50]. However, such a model still trains the hidden layer of the autoencoder on corrupted data, and the feature quality of the hidden layer may still be influenced by training data with a large fraction of corruptions.

Our model isolates noise and outliers in the input, and the autoencoder is trained after this isolation. Thus, our method promises to provide a representation at the hidden layers which is more faithful to the true representation of the noise-free data.

4.2 Hierarchical Robust Deep Autoencoders

The HRDA model has been inspired by the idea of a stacked denoising autoencoder [60] that builds a hierarchical model based on multiple, similar structured components, and then proceeds to train them layer-wise. The stacked denoising autoencoder is a supervised classification algorithm in that it requires noise-free data during it pre-training phase and then needs to adapt to the ground truth in its fine-tuning phase. When a stacked denoising autoencoder pre-trains the weights

of its deep feed-forward neural network, each denoising autoencoder layer takes the hidden layer from the last layer as the input.

4.3 Robust Variational Autoencoders

A number of advanced applications of generative models have recently been proposed, which include generating plausible images from human-written descriptions [52, 63], recovering photo-realistic textures from heavily down-sampled images [37], changing one image to be more like the other, as well as, generating novel imagery from rough user drawings [70]. Building good generative models of realistic samples has become a fundamental requirement of current AI systems [19, 20, 52].

There are two primary approaches to generative models, Generative Adversarial Networks (GANs) [26] and Variational Autoencoders (VAEs) [20, 31]. GANs train a generator that captures the data distribution and a discriminator that estimates the probability that a sample came from the training data, rather than the generator [26]. A VAE assumes that a collection of latent variables generates all the observations [20, 31]. Recently, various flavors of GANs and VAEs have been proposed, which have achieved compelling results in the image generation area. Danilo J. Rezende et al. [54] combined deep neural networks with approximate Bayesian inference to derive a directed generative model. Deep Convolutional GANs [51] first introduced a convolutional architecture in GANs which significantly improves the visual quality. More recently, David Berthelot et al. [4] provided Boundary Equilibrium Generative Adversarial Networks (BEGAN) with a new approximate convergence measurement, improving the speed and the stability of loss training.

Recent proposed research with generative models either focus on removing noise from corrupted input only [17, 58, 64] or generating new images from available cleaned data, which can be obtained from existing off-the-shelf denoising methods [10, 30].

5 Robust Deep Autoencoders

5.1 Robust PCA and Deep Autoencoder Combination

In [69], we proposed novel extensions to deep autoencoders which not only maintain a deep autoencoders' ability to discover high quality, non-linear features but can also eliminate outliers and noise *without access to any clean training data*. As a combination of deep autoencoders and Robust PCA, we name our model *Robust Deep Autoencoders (RDA)*. An RDA inherits the non-linear representation capabilities of autoencoders combined with the anomaly detection capabilities of RPCA. The key insight is that noise and outliers are essentially incompressible and therefore cannot effectively be projected to a low-dimensional hidden layer by an autoencoder. In particular, if exceptions could be allowed to the autoencoder loss function, then the low-dimensional hidden layer could provide accurate reconstruction, with a low-dimensional hidden layer, *except for those few exceptions*. Just as in RPCA, when the noise and outliers are isolated, the remaining data can be accurately reconstructed by an autoencoder with such a low-dimensional hidden layer.

Inspired by RPCA, we augment autoencoders with a filter layer. The filter layer culls out the anomalous parts of the data that are difficult to reconstruct, and the remaining portion of the data can be represented by the low-dimensional hidden

layer with small reconstruction error. An RDA also splits input data X into two parts $X = L_D + S$, where L_D represents the part of the input data that is well represented by the hidden layer of the autoencoder, and S contains noise and outliers which are difficult to reconstruct. By removing the noise and outliers from X, the autoencoder can more perfectly recover the remaining L_D. To achieve this property, our loss function for a given input X could be thought of as the ℓ_0 norm of S, which counts the number of non-zero entries in S, balanced against the reconstruction error of L_D, as in the optimization problem

$$\begin{aligned} \min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_0 \\ \text{s.t. } X - L_D - S = 0, \end{aligned} \tag{11}$$

where $E_\theta(\cdot)$ denotes an encoder, $D_\theta(\cdot)$ denotes a decoder, and S captures the anomalous data, L_D is a low dimension manifold and λ is a parameter that tunes the level of sparsity in S. *λ plays an essential role in our analysis.* In particular, a small λ will encourage much of the data to be isolated into S as noise or outliers, and therefore minimize the reconstruction error for the autoencoder. Similarly, a large λ will discourage data from being isolated into S as noise or outliers, and therefore increase the reconstruction error for the autoencoder.

We make extensive use of the λ parameter in our numerical results. In particular, one can tune λ depending on the desired use of the robust autoencoder. For example, λ can be tuned to maximize the performance of the features provided by the hidden layer in some supervised learning problems or λ can be tuned to optimize false alarm rates in unsupervised anomaly detection problems.

Unfortunately, the optimization problem in (11) is not computationally tractable. However, following the RPCA literature [15, 21, 48], one can relax the combinato-

rial $\|S\|_0$ term of the optimization (12) by replacing it with a convex relaxation $\|S\|_1$, giving rise to the optimization

$$\begin{aligned} \min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_1 \\ \text{s.t. } X - L_D - S = 0. \end{aligned} \tag{12}$$

In the constraint of (12) we split the input data X into two parts, L_D and S. L_D is the input to an autoencoder $D_\theta(E_\theta(L_D))$ and we train this autoencoder by minimizing the reconstruction error $||L_D - D_\theta(E_\theta(L_D))||_2$ through back-propagation. S, on the other hand, contains noise and outliers which are difficult to represent using the autoencoder $D_\theta(E_\theta(\cdot))$. *$\lambda$ plays an essential role in our analysis.* In particular, a small λ will encourage much of the data to be isolated into S as noise or outliers, and therefore minimize the reconstruction error for the autoencoder. Similarly, a large λ will discourage data from being isolated into S as noise or outliers, and therefore increase the reconstruction error for the autoencoder. The objective $||L_D - D_\theta(E_\theta(L_D))||_2$ does not specify any particular form for the encoding and decoding pair E and D, herein we follow the standard practice of having $E_\theta(x) = E_{W,b}(x) = \text{logit}(W \cdot x + b_E)$ and $D_\theta(x) = D_{W,b}(x) = \text{logit}(W^T \cdot x + b_D)$ [34].

In the constraint of (12) we split the input data X into two parts, L_D and S. L_D is the input to an autoencoder $D_\theta(E_\theta(L_D))$ and we train this autoencoder by minimizing the reconstruction error $\|L_D - D_\theta(E_\theta(L_D))\|_2$ through back-propagation. S, on the other hand, contains noise and outliers which are difficult to represent using the autoencoder $D_\theta(E_\theta(\cdot))$.

5.2 Anomalous Feature and Instance Detection

Our basic RDA in (12) assumes that the noise and outliers are unstructured, and thus we have used a generic ℓ_1 penalty on S to introduce element-wise sparsity. However, in many cases we may have anomalies that are structured.

In particular, we can view our training data as a matrix where each row is a data *instance*, such as a picture in an image processing application or a packet in a network analysis application, while each column is a *feature*, such as a pixel in an image or a particular byte in a packet. An example of such a data matrix is shown in Figure 2.

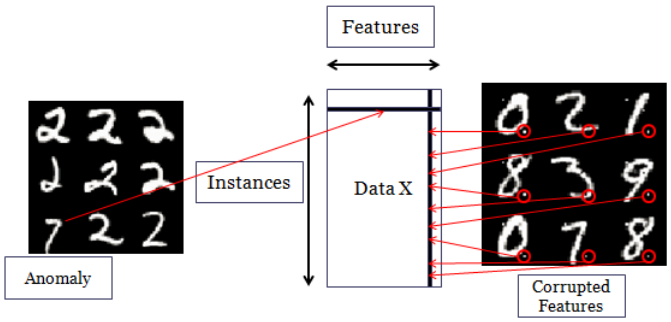

Figure 2: This figure shows two examples of group anomalies. For example, on the left, we see a collection of 2s corrupted by a single 7. This corresponds to a row of X being anomalous. Similarly, on the right, we see a collection of digits where a particular pixel is always on. This corresponds to a column of X being anomalous. Our structured $\ell_{2,1}$ penalty is intended to address both of these cases.

Accordingly, we treat two different types of *group anomalies*. First, we have the case where a particular feature is corrupted across many instances. For example, perhaps a digital camera has a bad pixel in its image plane. Second, we have the case where a particular instance is anomalous across many different features.

For example, one may have a picture of a 7 mixed into a group of pictures all of which are otherwise pictures of 2s, as in Figure 2. As a consequence, we have developed additional techniques that group errors across elements in S to enhance anomaly detection. In particular, both cases can easily be treated by our methods by generalizing the ℓ_1 penalty to a grouped $\ell_{2,1}$ norm [6].

In particular, the $\ell_{2,1}$ norm can be defined as

$$||X||_{2,1} = \sum_{j=1}^{n} ||x_j||_2 = \sum_{j=1}^{n} (\sum_{i=1}^{m} |x_{ij}|^2)^{1/2} \tag{13}$$

and, it can be viewed as inducing a ℓ_2 norm regularizer over members of each group and then a ℓ_1 norm regularizer between groups [6]. Similarly, if we want to group the elements of a row and then look for sparse anomalies among the collection of rows, we merely need to look at the transpose of X and use $||X^T||_{2,1}$.

With definition (13) in mind, we can pose the following optimization problem

$$\begin{aligned} \min_{\theta,S} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda \|S\|_{2,1} \\ \text{s.t. } X - L_D - S = 0, \end{aligned} \tag{14}$$

where the terms are defined as in (12) except the ℓ_1 norm has been replaced by the grouped norm $\ell_{2,1}$.

If we assume the anomaly instance exists in our data , for example in Figure 2, the element-wisely removable of noise may not be sufficient evidence to claim anomalies, and single element noise does not equal to anomalies. As a consequence, we group errors across rows in S to enhance anomalies detection. If an error of one group is larger than the threshold, this group is highly likely to be an anomaly. Such tasks require binding elements through rows to consider row-wise blocks as an ensemble of anomalies to consider instances as anomalies. We select

anomalies based these

Similarly, if we are interested in analyzing data with anomalous data instances, then we merely need to consider the following optimization problem using the grouped penalty $\ell_{2,1}$ on S^T:

$$\begin{aligned} \min_{\theta,S} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S^T\|_{2,1} \\ \text{s.t. } X - L_D - S = 0. \end{aligned} \tag{15}$$

In particular, (15) can be thought of as finding anomalous instances that are hard for an autoencoder to reconstruct based upon the other instances in the training data. For example, in an image processing application, one may have many pictures of cars with a few pictures of dogs interspersed among them.

This optimization problem says we measure the reconstruction error of each instance. The majority instances are similar, and they contain information redundancy. Such majority instances can be inferred from each other. While the anomalies stand out, they share less information with the majority. As a consequence, anomalies give higher reconstruction errors than the majority. By alternatively reconstruct L_D and shrink S, we will get high error instances in S and they are claimed as anomalies.

5.3 Experimental Evaluation

We test our RDA models on the well-known image recognition MNIST data set [36].The training set consists of $50{,}000$ instances, and each instance is a 28×28 pixel image. Figure 3 shows a quick view of uncorrupted data.

Figure 3: This figure shows an example of uncorrupted MNIST data.

5.3.1 Denoising RDA with ℓ_1 norm

In Figure 4, we show the key results of a denoising task for our proposed ℓ_1 penalized Robust Deep Autoencoder as compared against a standard autoencoder.

Our experiments consists of taking the digits from the MNIST data set and corrupting them with various levels of noise. In particular, we randomly pick a certain number of the pixels for each instance, and we set the pixel value to 0 if the original pixel value is larger than 0.5, and we set the pixel value to 1 otherwise. These corrupted images are used to train a normal autoencoder and a RDA each with exactly the same number and breadth of layers. To judge the quality of the features produced by our hidden layers we use the following procedure. In particular, we use the values in the hidden layer as features for a supervised classifier. For a fixed supervised classifier, we presume that higher test error rates are indicative of lower feature quality.

Results of comparing standard autoencoders are concisely summarized in Figure 4, where the x-axis shows different corruption levels, and the y-axis shows different λ values in RDAs. Red area in Figure 4 indicates where the error rates

of the RDA are superior to those of the normal autoencoder, and blue indicates where the converse is true. The errors rates are bases upon the prediction of image labels with respect to different degrees of corruption and different values of λ in the Robust Deep Autoencoders. As can be seen, the images range from very modest corruptions to those in which the original digits cannot be seen. Most importantly, the area marked by ② shows a large region, covering many different levels of corruption and values of λ, in which the RDA performs better than the standard autoencoder on the image classification task. In particular, for some levels of corruption, and values of λ, the robust autoencoder performs up to 30% better. On the right, we show several examples of images from the red area.

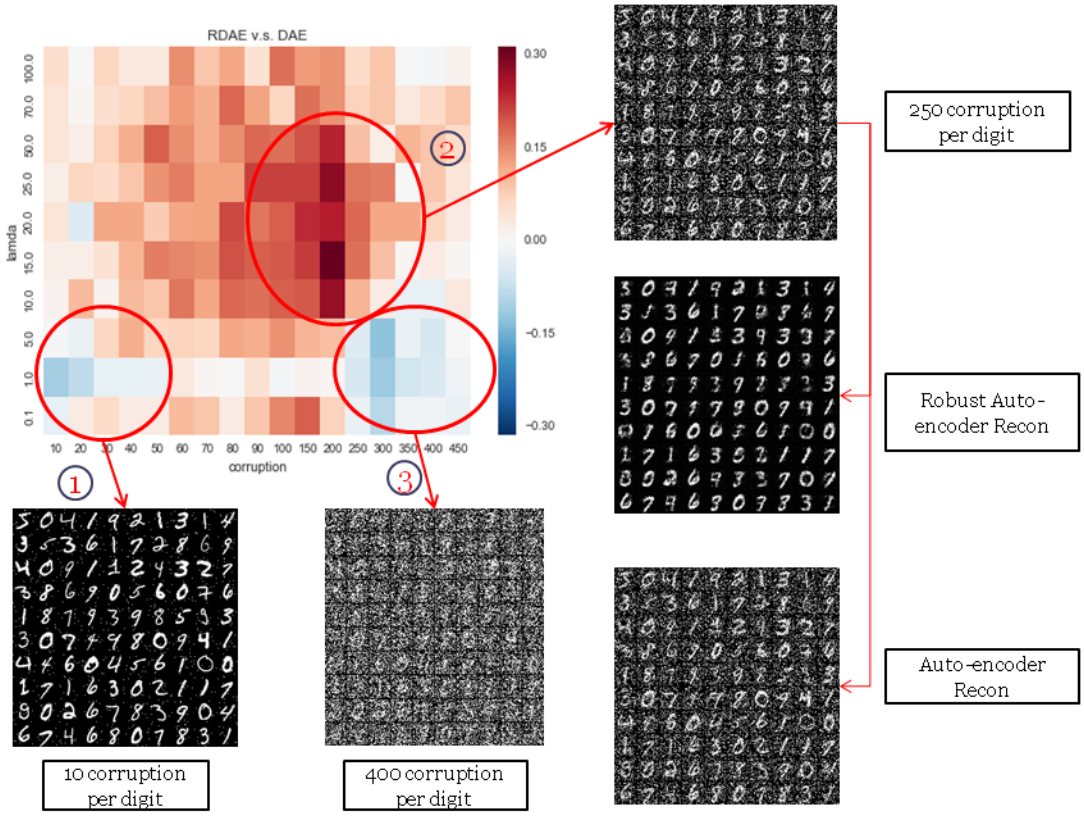

Figure 4: This figure shows the difference between error rates for the features constructed by a normal autoencoder and an RDA. The two images on the bottom, indicated with ① and ③, show examples of images with the designated amount of corrupted pixels, where the input images range from very modest corruptions to those in which the original digits cannot be seen. Most importantly, the area marked by ② shows a large region, the robust autoencoder performs up to 30% better. As can be seen on the right, the original images on top are quite corrupted, while the images in the middle, produced from the RDA's output layer, are largely noise free. However, the standard autoencoder, as it only has noisy imagery on which to train, faithfully, but inappropriately, reproduces the noise.

In the Figure 4, from left to right, the input X has a larger and larger fraction of corrupted pixels. As specified by our noise model, the number of corrupted pixels ranges from 10 to 350 (out of 784) per image. From bottom to top, the λ value grows from 0.1 to 100. In particular, we can see the Robust Deep Autoencoders and normal autoencoders get similar test errors when there are few corruptions, e.g. 10 to 50 corrupted pixels for each image (as shown in Figure 4 by case ①). This fact should not be surprising, since for such images the ℓ_1 penalty does not play a pivotal role.

However, when the noise increases, for example in the cases of area ② in Figure 4, and one has from 80 to 300 corrupted pixels per image, and *the normal autoencoder has up to* 30% *higher error rates than the Robust Deep Autoencoders!* The red areas in Figure 4 indicate those experiments where noise has significantly reduced the feature quality provided by the autoencoder's hidden layer, while the Robust Deep Autoencoder's hidden layer is immune to the noise, due to S and the ℓ_1 penalty. Also, from a more qualitative point of view, one also finds that the reconstructed images from the Robust Deep Autoencoders are *cleaner* digits and might be more desirable for consumption by a human analyst. Finally, when the fraction of corrupted pixels continues to grow, say about above 300 corrupted pixels per image, in the cases of area ③ in Figure 4, neither model can produce high-quality features and the testing accuracy of both methods is again the same.

5.3.2 Outlier Detection with $\ell_{2,1}$ norm

Our anomaly detection experiments begin by gathering images of the digit "4" from the MNIST dataset, and these images will comprise our nominal data. This nominal data is then corrupted by mixing in images which are randomly sampled

from other digit's images (e.g. "0", "7","9" etc.). The mixed data contains 4859 nominal instances and 265 anomalies. Accordingly, the ratio of anomalies to total number of instances in this set is about 5.2%.

Figure 5 shows a set of examples that are intended to provide the reader within intuition as to how λ influences the predictions and sparsity of S when using the $\ell_{2,1}$ norm.

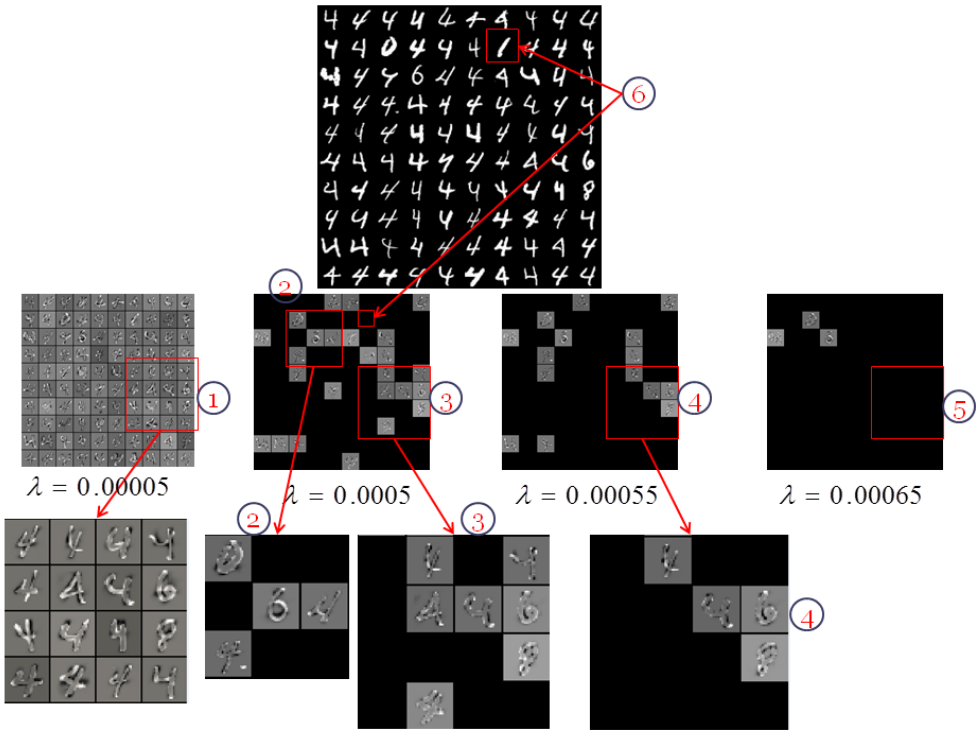

Figure 5: This figure shows how the sparsity of S changes with different λ values. A small λ places a small penalty on S, and the RDA emphasizes minimizing the reconstruction error by marking many images as anomalous and giving rise to many false-positives. λ then can be increased to trade-off false-positives for false-negatives. The optimal λ should balance both false-positive and false-negative rates. Thus, we use the F1-score to select the optimal λ.

As shown in Figure 5 our experiment proceeds as follows. λ is used to control the sparsity of S. In particular, a small λ places a small penalty on S, and the RDA emphasizes minimizing the reconstruction error by marking many images as anomalous and giving rise to many false-positives. λ then can be increased to trade-off false-positives for false-negatives. In Figure 5, ① shows a small λ, namely $\lambda = 0.00005$, which levies a small penalty on S, and the robust autoencoder emphasizes minimizing the reconstruction error by marking many images as anomalous. Accordingly, S is dense and every instance is non-zero. Since non-zero rows in S are marked as anomalies, the RDA has a high false-positive rate. In the case of ②, ③, and ④ the λ value is larger, placing a heavier penalty on S, and forcing S to be sparser. Many rows are shrunk to zero which reduces the false-positive rate, but also increases the false-negative rate. For example, ⑥ indicates an example of a false-negative; a "1" digit is supposed to be picked out as an outlier, but is marked as nominal. When the λ value gets even larger, as in ⑤, the large penalty on S causes every row of S to be shrunk to zero. In this cases, S is the zero matrix and thus no instance will be marked as anomalies. Accordingly, the optimal λ should balance both false-positive and false-negative rates. Thus, we use the F1-score to select the optimal λ.

To validate the performance of our outlier and anomaly detection methods, we compare our model against a state-of-the-art outlier detection method, namely the Isolation Forest [39]. The key idea of Isolation forests is that anomalies are 'few and different', which make them more susceptible to isolation than normal points [39].

We use the Isolation Forests model as implemented by version 0.18.1 of the scikit-learn package [49]. The isolation forest model take two variables: the num-

ber of trees and the fraction of outliers. We fix the number of trees to 100 and optimize over the outlier fraction from 0.01 to 0.69 (similiar to the optimization of λ in the RDA). We pick the best fraction number based on the F1-score.

We compare the $\ell_{2,1}$ RDA and isolation forest on their best F1-score and on how they perform across a range of parameters. From Figures 6 and 7, we can see the RDA gets an F1-score of 0.64 with its optimal λ parameter, while the highest F1-score achieved by the Isolation Forest is 0.37, which is a 73.0% improvement.

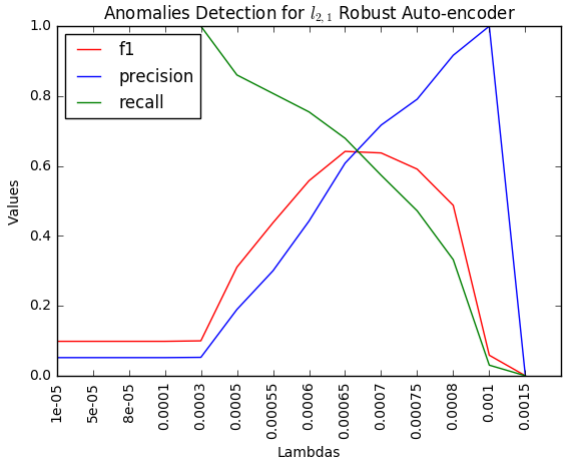

Figure 6: This figure shows the precision, recall and F1-scores with different λ values for the RDA. In the cases of a small λ, every row is non-zero and all the rows are marked as anomalies. We get high recall scores but very low precision and F1-scores. As the λ values increase, the F1-score and score increase reaching a maximum at $\lambda = 0.00065$ with an F1-score of 0.64.

5.4 Section Summary

In this section, we have shown how denoising autoencoders can be generalized to the case where no clean, noise-free data is available, creating a new family of methods that we call"Robust Deep Autoencoder". These methods use an anomaly regularizing penalty based upon either ℓ_1 or $\ell_{2,1}$ norms. We also extend the way of denoising of RDA using ℓ_1 to detect structured anomalies using $\ell_{2,1}$ norms, which re-

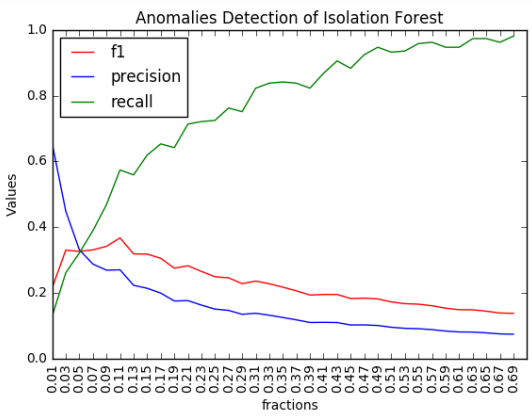

Figure 7: This figure shows the precision, recall and F1-score as we vary the fraction ratio for the Isolation Forest algorithm from 0.01 to 0.69. The optimal F1-score that the Isolation Forest achieves is about 0.37 when the fraction is equal to 0.11. This F1-score is approximately 73.0% worse than the score achieved by the RDA.

sults a novel outlier detection method. We have shown the superior performances of our proposed methods comparing to the state-of-the-art existing models.

6 Robust Sparse Autoencoders

6.1 Robust Sparse Autoencoders

In this section, we will detail a novel usage of the $X = L + S$ framework which leverages an $\ell_{2,1}$ regularization term on *hidden* layers and gives rise to more concise and informative hidden features. In particular, our proposed model is based on $\ell_{2,1}$ sparse autoencoders, and we derive "Robust Sparse Autoencoders (RSA)" from the RDA by replacing the normal deep autoencoder in equation 12 by a sparse autoencoder. RSA demonstrate the flexibility of the basic RDA model that the normal deep autoencoder part can be replaced by more complicated autoencoders to discover different types of low-dimensional and non-linear manifolds. The $\ell_{2,1}$ norm regularization in $\ell_{2,1}$ sparse autoencoders pursue sparse representation of feature

on the hidden layers, and thus they give more concise and lower dimensional representations.

With the $\ell_{2,1}$ sparse autoencoder defined in (7), a robust sparse autoencoder is therefore defined as:

$$\begin{aligned} \min_{\theta,S} \|X - D_\theta(E_\theta(L_D))\| + \beta\|E_\theta(L_D)\|_{2,1} + \lambda\|S^T\|_{2,1} \\ \text{s.t. } X - L_D - S = 0, \end{aligned} \tag{16}$$

where the various terms are the same as in (7) and (12). Actually, we consider $\|X - D(E(L_D))\|$ and $\beta\|E(L_D)\|_{2,1}$ as a single term, where a sparse autoencoder in which the reconstruction cost $\|X - D(E(L_D))\|$ and regularization term $\beta\|E_\theta(L_D)\|_{2,1}$ work together to find a suitable low dimension representation. Minimizing them encourages data put into S that causes the remaining L to not be low-dimensional. Minimizing the $\lambda\|S^T\|_{2,1}$, on the other hand, pushes data away from S and into the other terms. Alternating minimization results in a local best split of $X = L+S$. The β and λ values control the weights of penalty terms and tuning it will essentially influence the splits of $X = L + S$. In particular, β is the weight of regularization on hidden layers that a large β causes the hidden layers to be low-dimensional, while a small β allows the hidden layers to be high-dimensional. Such β, acting as a "dimension penalty weight", also decides the split of X. Intuitively, low-dimensional hidden layers requires that L_D represent a small portion of X and much of the data is isolated into S. On the other hand, high-dimensional hidden layers will represent more data from X and leave little data in S. On the other hand, λ controls the splits of X controversially that a large λ discourages much of the data to be isolated into S, and therefore L_D take much of the data that requires more hidden features to be represented accurately.

There are some extreme cases that bear special attention. For example, when β is zero, a robust sparse autoencoder (16) is equivalent to a robust autoencoder (12). When λ is zero, a robust sparse autoencoder (16) works identically as a sparse autoencoder. When both β and λ are zero, a robust sparse autoencoder (16) is as same as a normal autoencoder. When β goes to infinity, a robust sparse autoencoder (16) is not able to reduce the dimension of the data and thus it fails to detect any anomalies. When λ goes to infinity, a robust sparse autoencoder (16) is a sparse autoencoder.

6.2 Experimental Evaluation

As an extension model, we test RSAs using the same data as we train the RDA, detailed description about data can be seen in Section 5.3. Figure 8 presents full sights of our experiments on RSAs, the sparsity of S is influenced by both β and λ. Since non-zero rows in S will be marked as "outliers", sparsity of S results in different F1-score on the labeled data. When λ increases, the recall decreases but precision increases. As a consequence, with each fixed β, λ changes along rows. For each row, the F1-score of RSA increases at the beginning of λ increasing and falls after turning points. While β impacts RSA in a controversial way that a large β reduces "tolerance" of the majority and more instances will be suspected to "outliers". As a consequence, a large β has a high false-positive rate and a low false-negative rate. On the contrary, RSA with a small β generates high precision-scores but low recall-scores. A large β yield low precision-scores but high recall-scores. The F1-score gets its highest value when precision and recall are balanced. In Figure 8, we show the F1-score changing with β and λ. The F1-score reaches the maximum of 0.36 at $\lambda = 6.5$ and $\beta = 0.08$.

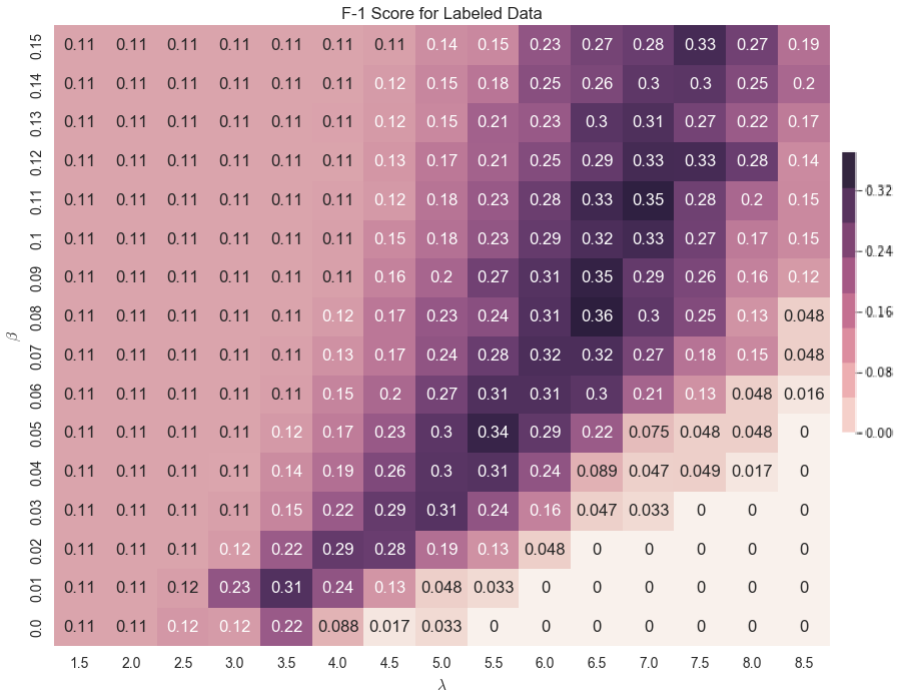

Figure 8: This figure shows how F1-score changes with different λ and β values in the robust sparse autoencoder model. In the bottom rows, RSA with small λs returns us high recall-score but low precision-score. From this picture, one can find that F1-score increases at the beginning of increasing β because of recall-score increasing dramatically while fall to decrease for precision-score dropping severally.

6.3 Section Summary

RSA is an extension of RDA which demonstrates how RDA can augment and extend any advanced autoencoder with the ability to detect anomalies. Classic sparse autoencoders [46] and our proposed $\ell_{2,1}$ sparse autoencoders are such examples of advanced autoencoders which are faithful to classic non-linear dimension reduction which detect a consistent set of mapped features across all training data. In particular, this model provides a sparse autoencoder which is more faithful to clas-

sic dimension reduction in that it identifies a consistent set of important features across all training data. We demonstrate how this $\ell_{2,1}$ sparse autoencoder can also be used in the RDA framework discussed above. Combining sparse autoencoders and the idea of robust splitting, RSA offer a method that keep the low-dimensional non-linear manifolds impregnable to the anomalies. We demonstrate the superior performances of our proposed methods on a selection of benchmark problems.

7 Hierarchical Robust Deep Autoencoders

7.1 Hierarchical Robust Deep Autoencoder

Inspired by stacked denoising autoencoder [24] that hierarchical structures are able to enhance single layer's neural network models, we propose a model consisting of a hierarchical collection of RDAs. This hierarchy, in which the building blocks of a deep network are themselves deep neural networks, has inspired us to call our new model a "Hierarchical Robust Deep Autoencoder (HRDA)". By using this hierarchical construction, we demonstrate how the effectiveness of RDAs can be improved for both anomaly detection and feature representation. In addition, the framework that we have developed for HDRAs allows several other direct extensions by allowing many generalizations of a standard autoencoder to be used in the same RDA framework.

Our idea of building an "Hierarchical Robust Deep Autoencoders (HRDA)" where each layer in the HRDA is an individual RDA, and the connections between layers are the L part of data which is the output of the last layer. In this section, the term "a layer" refers to a single RDA which does not imply the autoencoder part of each RDA being shallow, single layer autoencoder. On the contrary, RDAs

usually use deep autoencoders. Each layer takes the input and isolates noise and outliers into S, and the remaining L is the input to the next layer. Multiple layers of RDAs accumulate the anomaly detecting effects of all layers. The L of the last part is the cleanest data and the Ss of all layers are detected anomalies.

The HRDA is flexible enough to handle both denoising tasks and detecting outlier tasks. If a HRDA consists of basic ℓ_1 regularized RDA in equation (12), it is targeted at then denoising tasks. Detecting outliers can be done by just changing its components to an $\ell_{2,1}$ regularized RDA in equation (15). The HRDA is a pipeline that is trained greedily. Once a layer is trained and produces an L and S, they are fixed and are not trained by any back-propagated errors of subsequent layers. Intuitively, such a process allows multiple RDAs to identify anomalies individually except each RDA's attempt is based on the previous RDAs' outputs.

However, if the layers of an RDA share the same configurations, the second layer RDA, which takes as an input the L produced by the first layer can, optimally, have an identical output and input. In other words, the L passed to the RDA can be exactly represented since it was already represented by the previous RDA. We argue that identically configured RDAs will result in an identical result even though they may be initialized with different parameters. In HRDA, if the first optimal RDA isolates anomalies S, the remaining L will be perfectly reconstructed by the second RDA. To prevent identical input and output, an HRDA requires hyper-parameters to vary between layers. For example, variations can be in the number of hidden layers, hidden layer sizes, different λs or all of these. These variations prevent HRDAs from overfitting anomalies and increase the robustness of the entire HRDA. In our experiments, we observed that the *variation* of λ values from layer to layer controls the performances of an HRDA. In denoising experi-

ments, we use an incremental strategy for λ. A small λ on the early layers allow much of the entries filtered into S, and larger λ on the later layers allow later layers to be more strict in their detection of anomalies. In outlier detection experiments, we use a decremented λ strategy that shallow layers with large λ only pick out obvious outliers to prevent false-positive rates. The L part contains fewer and fewer outliers with processing going deep. Small λ on deep layers help identify indistinct outliers to reduce false-negative rates but at the cost of higher false-positive rates.

7.2 Experimental Evaluation

7.2.1 HRDA for Denoising

By using ℓ_1 norm on each layer of RDA, an HRDA is allowed to detect noising in the input data. The setting of our denoising experiment is close to the denoising experiment of RDAs in Section 5.1 except we use the additive Gaussian noise and the number of corrupted pixels on each picture is much bigger than previous experiment. In Figure 9, we show the key results for our proposed ℓ_1 regularized HRDA as compared against a single RDA. In particular, the area indicated by the red rectangle covers a large region, covering many different levels of corruption and values of λ, in which the second layer of the HRDA performs better than the single RDA on the image classification task. This red area in Figure 9, demonstrates that, in the most of cases, the hierarchical structure helps eliminate noise. Most importantly, the area indicated by the red rectangle covers a large region, covering many different levels of corruption and values of λ, in which the second layer of the HRDA performs better than the single RDA on the image classification task. In particular, for some levels of corruption, and values of λ, HRDA performs

up to 28% better. However, if the λ is improperly chosen as a very small value, as indicated by the blue rectangle, both layers isolate informative digits into S and the remaining L produces worse results than that of the single RDA.

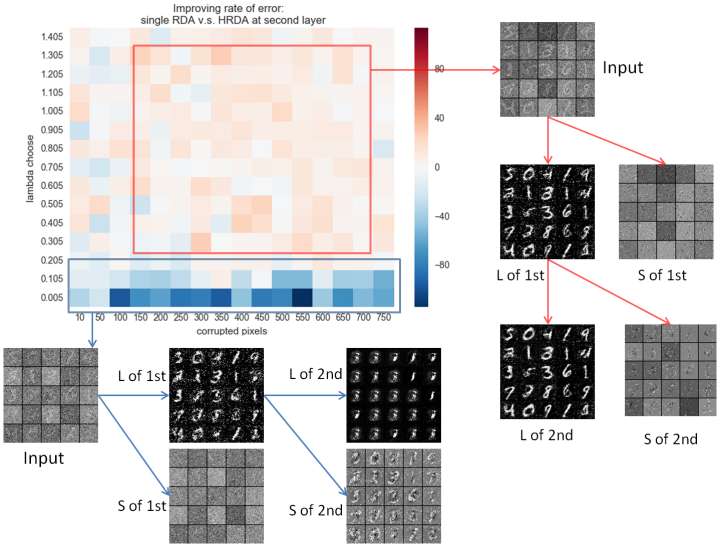

Figure 9: This figure shows the percentage of error rate improving for the features constructed by a RDA and the second layer of a HRDA. Red indicates where the error rates of the HRDA are superior to those of the single RDA, and blue indicates where the converse is true. The errors rates are based upon the prediction of image labels with respect to different degrees of corruption and different values of λ in the RDAs. The x-axis shows different corruption levels, and the y-axis shows different λ values. On the right, we show several examples of images from the red area. As can be seen, the original images on top are quite corrupted. Both the first and the second layers of the HRDA isolate certain amounts of noise, and the output L of the second layer is largely noise free.

7.2.2 HRDA for outlier detection

HRDA is allowed to detect outliers when $\ell_{2,1}$ RDAs form each layer. For the consistent convention, outlier detection experiments share the same settings with the outlier detection in the Section 5.2. As shown in Figure 10 our experiment proceeds as follows.

λ is used to control the sparsity of S. A small λ places a small penalty on S, and the RDA emphasizes minimizing the reconstruction error by marking many images as anomalous and giving rise to many false-positives. λ then can be increased to trade-off false-positives for false-negatives. Accordingly, the optimal λ should balance both false-positive and false-negative rates. Thus, we use the F1-score to select the optimal λ.

We choose a family of λ by decreasing each member with a factor of 0.65, since images are less corrupted and contain less outliers through each layers. Smaller λs allow subsequent more sensitive to abnormal patterns and isolate more instances into S. In Figure 10, the blue line is the first layer in the HRDA which is exactly as same as a single RDA. The green line indicate the F1-scores of the second layer. After the first layer filtering, the λ need to be small at second layer to detect outliers. With λ continually decreasing, the third layer indicated by the red line gets better result than the second layer. However, the fourth layer gets too small λ which results in it isolate everything into S and have high false-positive rate.

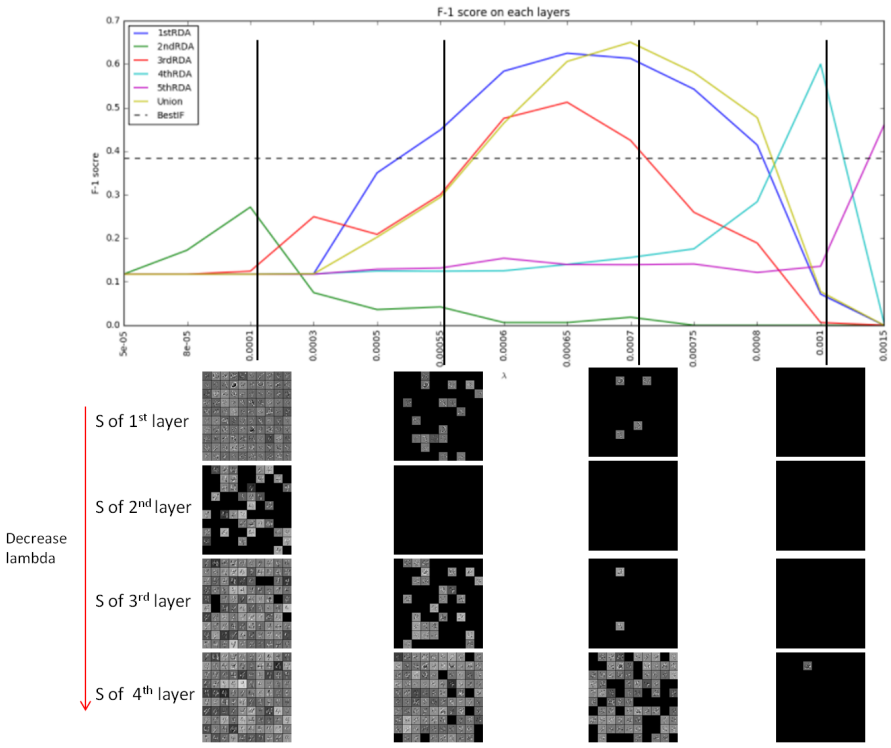

Figure 10: This figure shows the F1-scores with different λ values for each layer of the HRDA and union results of the first four layers. The F1-score of the first layer RDA reaches a maximum at $\lambda = 0.00065$ with an F1-score of 0.62. The black lines align S of each layer with its corresponding F1-score. Combing the first four layers, the united prediction result is indicated by yellow line, which reach the F1-score 0.65 when the starting λ is 0.0007.

We compare the $\ell_{2,1}$ RDA and Isolation Forest on their best F1-score and on how they perform across a range of parameters. From Figure 10, we can see that the single RDA gets F1-score of 0.62 with its optimal $\lambda = 0.0065$, and the HRDA with the union prediction gets the best F1-score with $\lambda = 0.007$. While the highest F1-score achieved by the Isolation Forest is 0.38. RDA has a 63.2% and HRDA has a 71.1% improvement.

7.3 Section Summary

In this work, we have shown how stacked denoising autoencoders can be generalized to detecting anomalies when there is no clean, noise-free data available, creating a new family of methods that we call *"Hierarchical Robust Deep Autoencoders (HRDA)"*. We provide an extension of Robust Deep Autoencoders (RDA) to hierarchical cases where an RDA forms each layer and it is trained in a greedy layer-wise fashion. Our HRDA models get improved results on both denoising and outlier detection tasks on MNIST dataset.

8 Robust Variational Autoencoders

This work is done by collaborating Yunmin Ren and Yun Yue and the material in this section is also described in [53].

Generative models, including Variational Autoencoders, aim to find mappings from easily sampled latent spaces to intractable observed spaces. Such mappings allow one to generate new instances by mapping samples in the latent space to points in the high dimensional observed space.

However, in many real-world problems, pervasive noise is commonplace and these corrupted measurements in the observed spaces can lead to substantial corruptions in the latent space. Herein, we demonstrate a novel extension to Variational Autoencoders, which can generate new samples without access to any clean noise-free training data and pre-denoising stages.

In this section, we provide details of another extension model, Robust Variational Autoencoders (RVAE), which build an anomaly filter onto normal variational autoencoders.

This work arises from Robust Principal Component Analysis (3) and Robust Deep Autoencoders (12), and we split the input data into two parts, $X = L + S$, where S contains the noise and L is the noise-free data which can be accurately mapped from the latent space to the observed space. Since our model enhances the robustness of Variational Autoencoders to noise, we name our model Robust Variational Autoencoder(RVAE). We demonstrate the effectiveness of our model by comparing it against standard Variational Autoencoders, Generative Adversarial Neural Networks, and other pre-trained denoising models.

The central idea of RVAE is that noise and clean data are essentially spawned from different distributions, and therefore generation of both noise and clean data from the same latent variables is highly unlikely. In particular, variational autoencoders assume all instances are generated from simple, low-dimensional distributions, but noise and anomalies share little information with clean data. This results in large errors if one tries to infer noise from generative mappings which are optimal on clean data.

We depict a brief structure of RVAE in Figure 11, where the noisy input is split into two pieces, L and S. L represents desired noise-free data, and it fits into a standard variational autoencoder that includes latent variables z, generative mapping $q(x|z)$ and inference mapping $p(z|x)$. While S is a filter that contains exceptions from the variational autoencoders of L and penalized by the ℓ_1 norm. We also provide a training algorithm for the splitting of L and S which is a non-differentiable and non-convex problem.

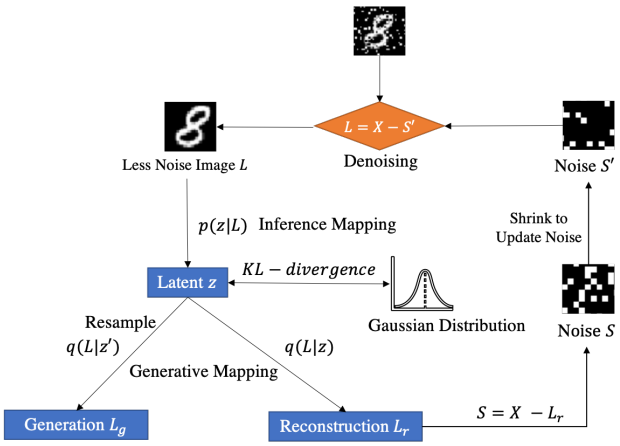

Figure 11: This figure shows the structure of RVAE. The noisy input is split into two pieces, L and S. L represents the clean data, and it is an input of a variational autoencoder, while S consists of noise and constrained by ℓ_1 norm.

8.1 Robust Variational Autoencoders

Since noise and normal data are essentially spawned from different distributions, the distributions of the normal data and the distribution of the noise in the latent space are also different. This key difference allows us to isolate noise from the normal data by way of augmenting the variational autoencoders with a filter layer. Such a filter removes the anomalous portion of the data which will mislead the latent variable distribution, and the remaining parts are faithful to the noise-free data. Similar to the RDA in (12), a Robust Variational Autoencoder splits the input data X into two parts $X = L + S$, where L represents the part of normal data that is well represented by a Gaussian distribution in the latent space, and S contains noise and outliers. To achieve this property, we pose the following RVAE optimization problem:

$$\begin{aligned} &\min_{\theta_1,\theta_2,L,S} \|L - D_{\theta_2}(E_{\theta_1}(L))\|_2 + KL(E_{\theta_1}(L)|\mathcal{N}(0,1)) + \lambda\|S\|_1 \\ &\text{s.t. } X - L - S = 0, \end{aligned} \tag{17}$$

where E_{θ_1} and D_{θ_2} represent inference mapping $q(z|x)$ and generative mapping $p(x|z)$ respectively, $\|L - D_{\theta_2}(E_{\theta_1}(L))\|_2$ represents a reconstruction error of the variational autoencoder part, and $KL(E_{\theta_1}(L)|\mathcal{N}(0,1))$ represents the Kullback-Liebler divergence (KL-divergence) [32], which measures differences between the distribution of the latent variables and a Gaussian distribution. The $\|S\|_1$ represents the ℓ_1 norm of S and λ is the key hyper-parameter. It controls the level of penalization on S and thus tunes the amount of data to be isolated as noise. A small λ encourages more of data to be isolated as noise. On the other hand, a large λ discourages data to be filtered into S. When λ is arbitrarily large then an RVAE is exactly the same as a VAE.

8.2 Experimental Evaluation

In this section, we demonstrate the effectiveness of our proposed RVAE by utilizing a standard benchmark data set, MNIST [35], and a direct drop-in replacement data set, Fashion MNIST [61]. Both data sets consist of $70\,000$ instances with 784 features each. Each instance of MNIST is labeled with an integer value between 0 and 9, while every instance of Fashion MNIST is comprised of 10-class fashion products. Rather than focusing on generating any specific classes' images, which is not the emphasis of this paper, we instead utilize all training samples without label information. As we will demonstrate in the next part, RVAE shows significantly superior performance when the model is trained on noisy, corrupted data. We corrupt original images with salt-and-pepper noise which, occurring as white and black pixels, is caused by sharp and sudden disturbances [57]. We also justify the robustness of model by testing Gaussian noise.

The salt-and-pepper noise corruption works by changing some randomly picked

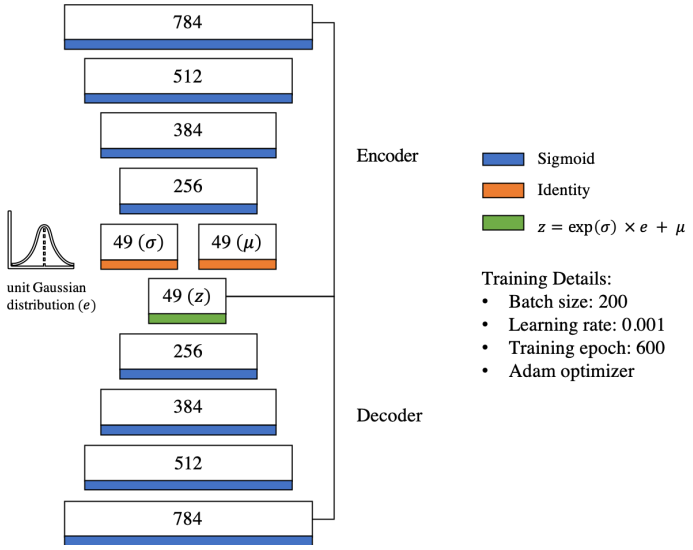

Figure 12: This figure shows the structure of VAE and the implementation details. RVAE, VAE, RPCA+VAE and LPF+VAE share the same structure.

pixels to 0 if the original values are larger than 0.5, and 1 if their values are smaller than 0.5. As for Gaussian noise, we add a value drawn from a Gaussian distribution with a scaling factor to every pixel. Then we clip each pixel between 0 and 1, i.e., we set the pixel value to 1 if the corrupted value is larger than 1 and 0 if it is smaller than 0. The first column of Table 16 shows instances of MNIST with 5% and 29% of the pixels corrupted by salt-and-pepper noise (first two rows), and Fashion MNIST with 9% and 41% of the pixels corrupted by salt-and-pepper corruption (last two rows). In the following section, we use slightly different percentages of corrupted images to train RVAE and other benchmark models.

8.2.1 Benchmark Methods

To evaluate denoising and generation abilities, we compare our model with two well-known generative models, traditional VAE [31] and GAN [26]. All the generative models share the same number of parameters such as the number of layers,

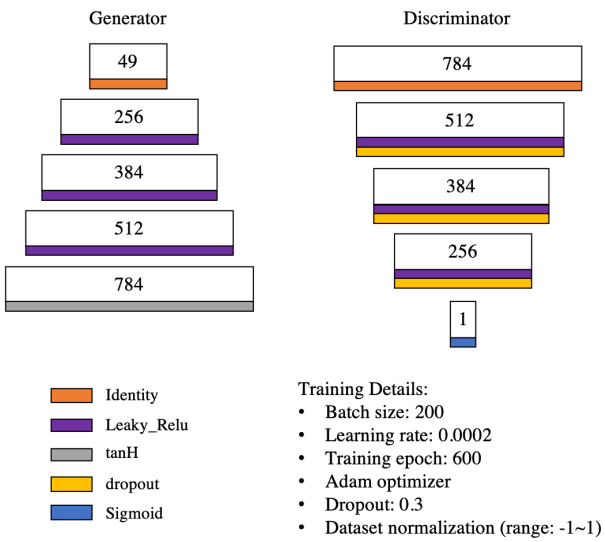

Figure 13: This figure shows the structure of GAN and its implementation details. GAN uses the same structure but different activation functions and different hyper-parameters.

dimensions of each layer, etc. Figure 12 and Figure 13 depict the basic structures of these two models. Other key hyper-parameters for reproducing results are listed below. The activation function used is sigmoid while leaky ReLU and TanH are used in the GAN. For other hyperparameters, all the models share the same learning rate with $1e-3$ while GAN uses $2e-4$. The batch size is chosen as 200 in each model. We train the VAE inside the RVAE for 20 epochs, with 30 iterations per epoch, to alternate projecting onto L_D and S, thus the number of iterations is 600. To keep the total number of iterations consistent with RVAE, the benchmark methods are also trained using 600 iterations.

Lastly, we also include a two-stages pipeline where we apply some well-known denoising model to remove the noise first, then use a standard VAE to generate images from the pre-processed images. Since VAE has shown the ability to generate well-quality images from clean images, the denoising method plays an important role that influences the final generating results. In particular, we pick two repre-

sentative denoising approaches: a Low-Pass Filter (LPF), which is a standard denoising method [10, 30], and RPCA, which is one of the inspirations of our RVAE model. As we introduced in Section 3.1, RPCA decomposes the input matrix X into a low-rank matrix L and a sparse matrix S. The low-rank matrix L is used as the cleaned images to generate new images. Our experiments show that although RPCA has the similar framework $X = L + S$ with our model, the *linear*, low-rank RPCA is not able to capture the complex structures of image data. It returns blurred filtered results from which generated images are less clear when compared with RVAE.

8.2.2 Evaluation Metrics

To judge the quality of images generated by the model, we use the "Fréchet Inception Distance" (FID) score, which is computed by considering the similarity of two distributions X_1 and X_2 [27]. FID score has been proven as an effective measure, which is consistent with human's visual inspection, to judge the quality of generated images [27]. It can also detect in-class data dropping, i.e. identical images will give bad FID score [40]. Mathematically, FID assumes that instances follow a continuous multivariate Gaussian, and its formula is:

$$\|\mu_1 - \mu_2\|^2 + Tr(\sigma_1 + \sigma_2 - 2\sqrt{\sigma_1 * \sigma_2}),$$

where (μ_1, σ_1) and (μ_2, σ_2) are the sample mean and covariance of X_1 and X_2. FID ranges from 0 to ∞ that a **small** FID score indicates a **high** similarity between X_1 and X_2 [27].

In our experiment, we calculate FID scores based on generated images and original noise-free images to measure the closeness between clean images and gener-

ated images where small FID scores indicate successful generations. Note, the FID score is only used for the final evaluation, and we never tune models according to the FID. These generative models are still unsupervised as they appear in the standard literature.

8.2.3 MNIST

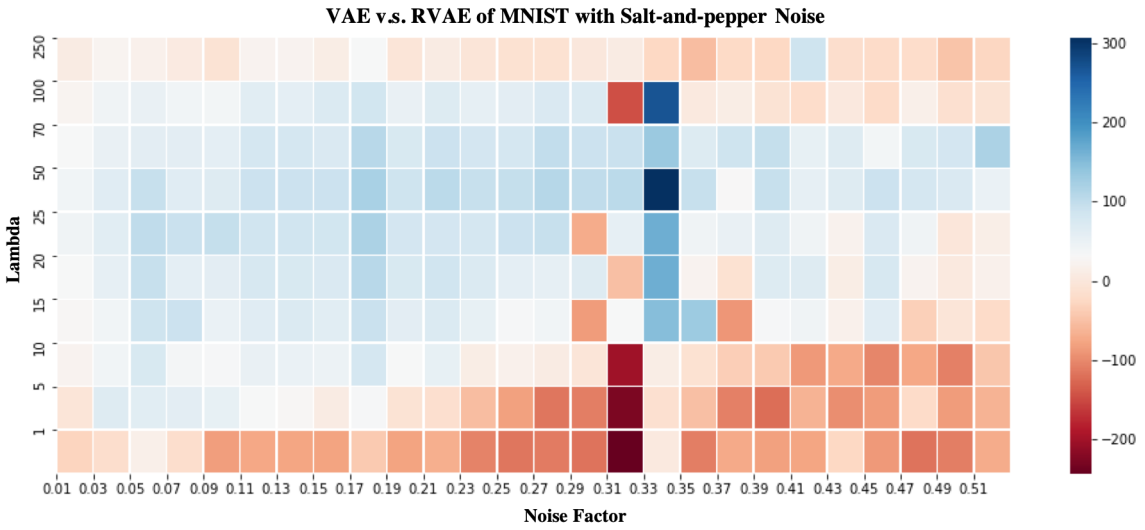

Figure 14: This figure illustrates the quantitative comparison of generation ability between VAE and RVAE when the same level of salt-and-pepper noisy images are used as input. The x-axis shows different corruption levels, while the y-axis represents different λ values. The blue area indicates the FID scores of RVAE are smaller than VAE's, meaning that the generation from RVAE is better than VAE, while the red area shows the opposite.

Figure 14 shows the quantitative comparison of generation ability between the VAE and the RVAE of which the input contains the same level of salt-and-pepper noise. Although both models are only trained with noisy images, the generated images from the RVAE have higher fidelity. In Figure 14, from bottom to top, the parameter λ from RVAE increases from 1 to 250. A small λ indicates a large number of pixels are isolated as noise, while a large λ means only a small part of data

belongs to S. From left to right, the ratio of corrupted pixels to the whole pixels in input images grows gradually from 1% to 51%. Each cell of the heatmap represents the difference of FID score between VAE and RVAE. Since the FID score has a negative correlation with visual quality check for the generated images [27], a *large* difference between VAE and RVAE indicates that the generation of RVAE is *better* than VAE. In particular, RVAE does not perform very well when λ is small, i.e. the value of λ is smaller than 10, since too much data is regarded as noise including the useful information. As λ increases, the penalty plays a more important role and less data are treated as noise in our model. When the value of λ is arbitrarily large then the RVAE is the same as the VAE. As can been seen from Figure 14, the difference between VAE and RVAE is small when the fraction of corrupted pixels is too small or too large. This is because both RVAE and VAE can generate good images when the input images are not corrupted too much while neither of them can produce high-quality images when the input data has too much noise. The blue areas show that RVAE is immune to the noise and can generate higher quality images than VAE due to the ℓ_1 norm of S. On average, our model shows **42.47%** improved image generation when the corruption ratio ranges from 3% to 27% and λ value varies from 10 to 100. The best result of our model shows **74.11%** improved image generation when the 33% pixels are corrupted (as shown in the darkest blue cell in Figure 14).

The generative examples of all the models we mentioned in Section 8.2.1 are appear in Table 16. In the first two rows, we provide images of 16 generated digits for each model when 5% and 29% pixels of the raw images are corrupted. These pictures show that RVAE has a strong capability to generate high visual quality digits, while the other models are unable to isolate noise or lose some information

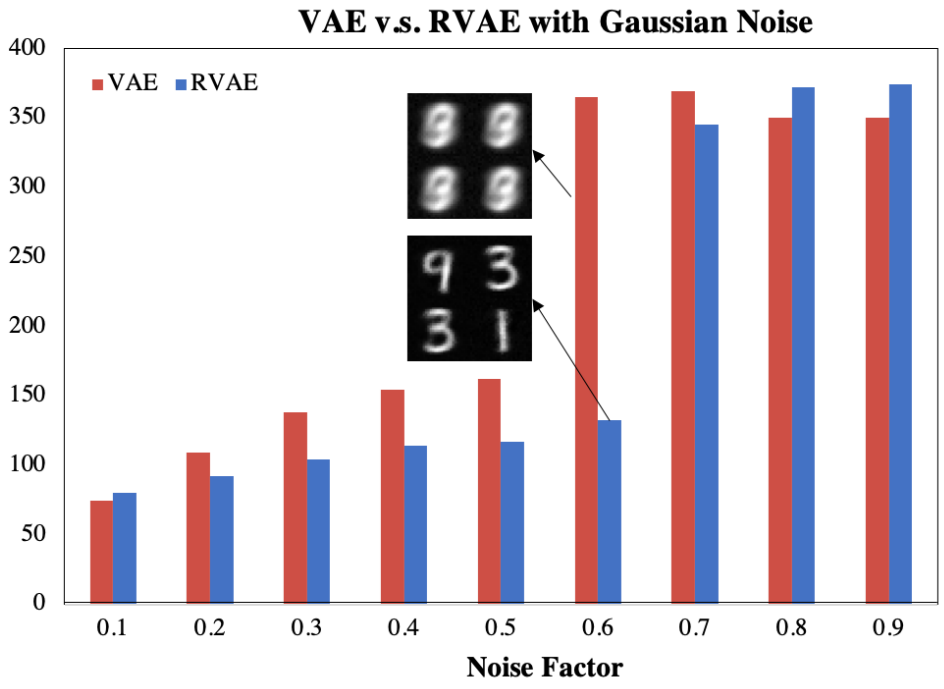

Figure 15: This figure shows the comparison of FID scores between VAE and RVAE. A small score indicates a high quality of generation. As one can see, RVAE works better than VAE when noise is not too large. Two examples of image instances are provided to show the generation abilities of VAE and RVAE. In particular, RVAE still can generate realistic images when the input images are corrupted with 60% Gaussian noise, while the generated images from VAE are unable to be identified when the input images are corrupted with the same level of noise.

when denoising and thus fail to generate clean and realistic images. In particular, we can see some noise is generated along with the new digits from VAE, while no meaningful images are generated by GAN. Although the two-stage models, RPCA+VAE and LPF+VAE, are capable of denoising and generating new images, the edge of each digit is less sharp when compared with RVAE. Some of the fake images generated by two-stage models are too dark and too blurry to be recognized, while the generation from RVAE is realistic and easy to identify.

To evaluate the robustness of our model, we also corrupt the images with Gaussian noise. As can be seen from Figure 15, RVAE gets a smaller FID score than VAE

when noise is not too large and there is a large difference between VAE and RVAE with 60% corruption. Two examples are provided to show the clear and blurry generation of RVAE and VAE respectively. As the noise increases to 70% and more, both RVAE and VAE cannot generate high quality images since too much corruption makes the input images unable to be recognized. Thus the generation of both models are hard to identify.

8.2.4 Fashion MNIST

To test the generative capability of RVAE, we implement our model with a more complicated data set, Fashion MNIST [61]. Table 1 illustrates the FID scores of Fashion MNIST with different levels of salt-and-pepper noise among different models. The smallest FID score at each noise level is marked in red color, indicating the best generation among our approach and all benchmark models. RVAE outperforms the other models in all cases except for 1% corruption case. It is because when the ratio of corruption is too small, RVAE may remove too much information and reduce the quality of generation. It is apparent from the table that RVAE can generate higher quality images than other benchmark models regardless of whether the noise level is small or large.

Noise	RVAE	VAE	GAN	RPCA+VAE	LPF+VAE
1%	113.39	93.80	612.09	174.84	172.16
5%	59.25	108.09	600.03	137.97	132.41
9%	68.70	122.55	708.67	137.48	146.17
13%	73.25	131.78	556.20	140.84	154.78
17%	82.43	144.22	732.73	144.75	164.76
21%	88.85	150.74	647.05	146.49	174.79
25%	93.62	159.41	596.36	151.92	182.61
29%	96.26	163.63	453.03	200.58	189.48
33%	102.42	170.86	463.21	161.19	193.26
37%	107.34	180.76	507.32	210.95	202.87
41%	111.24	189.72	563.85	165.00	208.20
45%	168.76	197.44	641.26	215.14	260.36
49%	120.78	213.99	652.82	172.54	216.86

Table 1: This table shows FID scores for Fashion MNIST with different levels of salt-and-pepper noise. The red entries indicate the best generation among our approach and all benchmark models.

More detailed examples are shown in the last two rows of Figure 16. When comparing with generative models, VAE can generate new image along with some noise, while GAN cannot generate anything meaningful. The quality of generation from RVAE is also better than the two-stage models, i.e. the edge of each fashion product is much sharper and clearer in RVAE's generation while the generated images from RPCA+VAE and LPF+VAE are blurry and unrealistic.

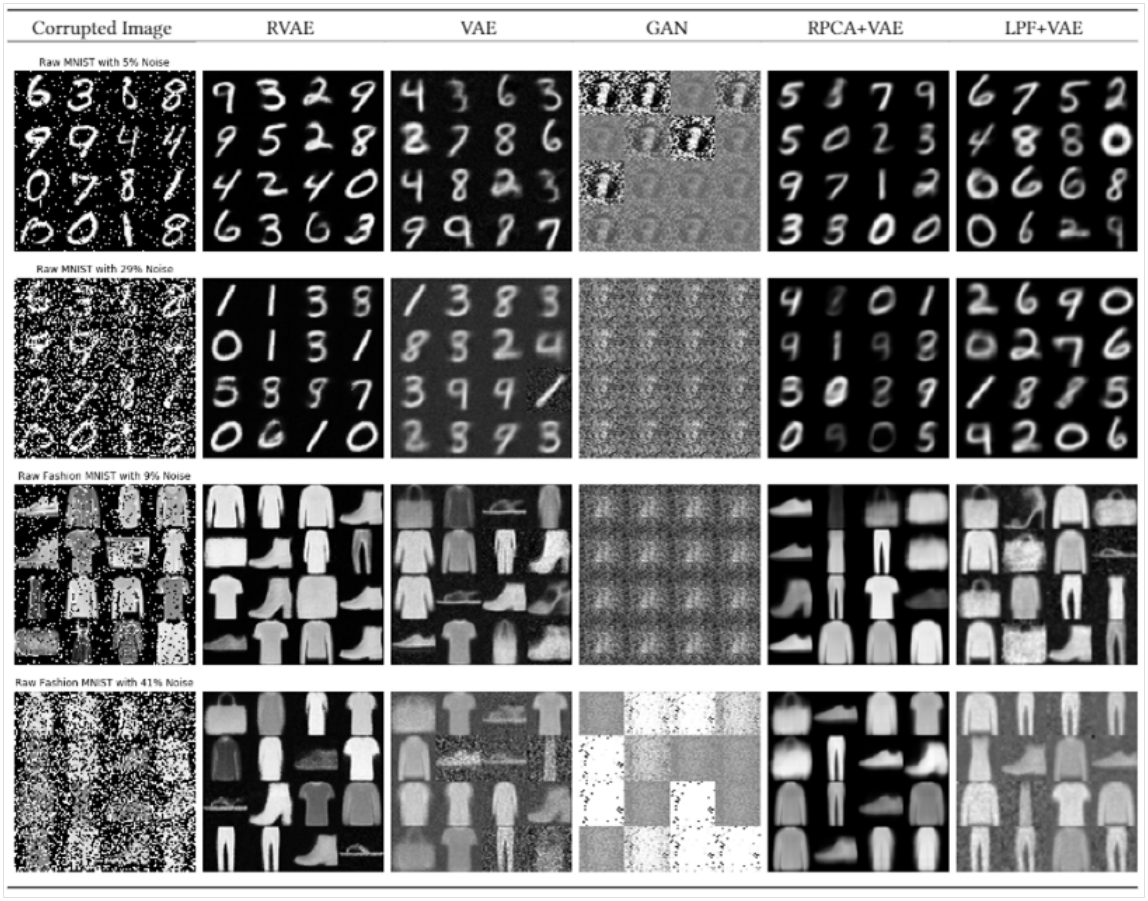

Figure 16: This table shows the generated examples from RVAE and other benchmark methods at different corruption levels and different data sets. The first column shows the input images with 5% and 29% corruption for MNIST and 9% and 41% corruption for Fashion MNIST.

8.3 Section Summary

In this work, Robust Variational Autoencoder reaches the gap between denoising and generation and shows that generate high quality images in the case where no clean, noise-free data is available. Same as previous sections, the framework $X = L + S$ plays a pivotal role to split noise and normal data, where L is the input to a standard variational autoencoder and S is regularized by ℓ_1 norm. We evaluate the effectiveness of our model with different data sets. The experiments show that

RVAE is faithful to its name, "robust", which, with a wide range of λ selection, generates reasonable images from different levels and types of noise of corrupted data. Also, we show that our integrated denoising-generative model has superior performance over separated denoisng and generation models.

Part II

Training Methods

9 Overview

In previous sections we have introduced a number of new methods. However, the standard training algorithm, back-propagation, is not efficient to solve the $X = L + S$ framework. In this part, we first present some ideas of background algorithms that our novel training algorithm is inspired by, including the Back-propagation, the Alternating Direction Method of Multipliers [7], and the Proximal Gradient [5]. We demonstrate how one can address such a problem by combining standard off-the-shelf solvers for optimizing the $X = L + S$ framework. However, since our full objective is not convex, the convergence of the method to a global minimum is non-trivial to guarantee. In particular, even the minimization of just the L part, cannot, in full generality, be guaranteed to and a global minimizer [8]. However, as we will demonstrate in Section 11, we have substantial empirical evidence that the optimization algorithm we present below provides high-quality solutions.

10 Background

10.1 Back-propagation

Back-propagation is a method used in most artificial neural networks to calculate partial derivatives that are needed in the computation of gradients to be used in training the network [25]. Back-propagation shows both the efficiency and general-

ization properties in training a neural network and has become a standard method in deep learning area [25]. Back-propagation computes an error at the output and propagates those errors back through the network's layers [25, 47, 56].

Back-propagation is a generalization of the delta rule to multi-layered feed-forward networks [47, 56], made possible by using the chain rule to iteratively compute gradients for each layer.

A typical back-propagation has two phases: 1. Forward propagation to compute the output of neural network and 2. Back propagation of errors to update the weights of the neural network. For example, a single layer in a neural network is often parameterized with the weights $W \in \mathbb{R}^{m \times n}$ and bias term $b \in \mathbb{R}^n$, and a activating function $\alpha(\cdot)\, \mathbb{R}^n \Rightarrow \mathbb{R}^n$. The forward propagation phase to compute the reconstruction cost [56]:

$$o \; = \; \alpha(W \cdot X \; + b)$$

Then, the second phase is to backward propagation of the cost to update the weight W and bias term b [56]:

1. Compute the gradient using the chain rule of calculus

$$\Delta_W \; = \; \frac{\partial \, o}{\partial \, \alpha} \frac{\partial \, \alpha}{\partial \, W}$$

$$\Delta_b \; = \; \frac{\partial \, o}{\partial \, \alpha} \frac{\partial \, \alpha}{\partial \, W}$$

2. Update the weights by the gradient

$$W^{l+1} \; = W^l \; - \; \lambda \cdot \Delta_W$$

$$b^{l+1} = b^{l} - \lambda \cdot \Delta_b$$

It was noted that back-propagation is not guaranteed to find the global minimum of the error function, but only a local minimum [47, 56]. However, in the range of this dissertation, it yields reasonable results supported by experiments.

10.2 Proximal Gradient

Proximal gradient methods are a generalized form of projection used to solve *non-differentiable* convex optimization problems. In particular, many interesting problems can be formulated as convex optimization problems of form

$$\min_{x \in \mathbb{R}^N} \sum_{i=1}^{n} f_i(x), \tag{18}$$

where f_i are convex functions defined from $f : \mathbb{R}^N \to \mathbb{R}$. Some of the functions f are *non-differentiable*, this rules out our conventional gradient-based optimization techniques like back-propagation in Section 10.1.

Proximal gradient methods proceed splitting, in that the functions $f_1, ..., f_n$ are used individually so as to yield an easily implementable algorithm [3, 9]. Iterative Shrinkage thresholding algorithm [3, 18] is one of proximal gradient method that is used to train the $\|\cdot\|_1$ and $\|\cdot\|_{2,1}$ norms in our model. The proximity operator of a convex function f at x is defined as

$$prox_f(x) = argmin_u \, (f(u) + \|x - u\|_2^2). \tag{19}$$

To solve ℓ_1 and $\ell_{2,1}$ norm optimization in our model, the f is set as $\|u\|_1$ and

$\|u\|_{2,1}$. Since both of them are convex and closed, there exist and is unique solution $prox_f(x)$, denoted as p, for all x. From optimally conditions of minimization [18], we get

$$p = prox_f(x) \rightarrow x - p \in \partial f(p), \tag{20}$$

where $\forall(x, p) \in \mathbb{R}^N \times \mathbb{R}^N$. (20) is a generalization form that allows existing more than one gradient for non-differentiable functions f [18]. If f is differentiable then (20) reduces to:

$$p = prox_f(x) \rightarrow x - p = \partial f(p), \tag{21}$$

More specifically, ℓ_1 and $\ell_{2,1}$ norms have differential form as

$$\frac{d(\|x\|_1)}{d(x)} = \begin{cases} -1, & x < 0; \\ 1, & x > 0; \\ 0, & x = 0. \end{cases} \tag{22}$$

(22) allows us to derive an algorithm that trains the S part in RDA model which we will detail in the next section.

10.3 ADMM

The alternating direction method of multipliers (ADMM) is a variant of the augmented Lagrangian scheme that uses partial updates for the dual variables [8, 9]. This method is often applied to solve problems such as [8]

$$\min_x f(x) + g(x).$$

This is equivalent to the constrained problem

$$\min_{x,y} f(x) + g(y),$$

$$\text{s.t. } x = y.$$

Though this change may seem trivial, the problem can now be attacked using methods of constrained optimization (in particular, the augmented Lagrangian method), and the objective function is separable in x and y [8]. The dual update requires solving a proximity function in x and y at the same time; the ADMM technique allows this problem to be solved approximately by first solving for x with y fixed, and then solving for y with x fixed. Rather than iterate until convergence, the algorithm proceeds directly to updating the dual variable and then repeating the process. This is not equivalent to the exact minimization, but surprisingly, it can still be shown that this method converges to the global minimum. Because of this approximation, the algorithm is distinct from the pure augmented Lagrangian method [8, 9].

Such methods are commonly used in convex optimization problems [7, 8, 9] and in such cases there is a body of theory that proves convergence and convergence rates of such methods [7, 9]. However, here we take a broader view and apply such methods in a non-convex context. The classic theorems do not apply in this case, however, we have substantial empirical evidence that our modified ADMM and the optimization algorithm we will detail below provide high-quality solutions.

11 Algorithm Training

In this section, we provide our idea and details to train for solving problems such as "Robust Deep Autoencoders" (12), (15) and "Robust Sparse Autoencoder"(16). In particular, we are inspired by ideas such as the Alternating Direction Method of Multipiers [8] and R. L. Dykstra's alternating projection method [9][1]. We iteratively optimize each of the terms in (12), (15) and (16) separately, interspersed with projections onto the constraint manifold. Since our full objective is not convex, the convergence of the method to a global minimum is non-trivial to guarantee. In particular, even the minimization of just the autoencoder $||L_D - D_\theta(E_\theta(L_D))||_2$ cannot, in full generality, be guaranteed to find a global minimizer [60]. However we have substantial empirical evidences that the optimization algorithm we present below provides high-quality solutions.

11.1 Alternating Optimization for Robust Deep (Sparse) Autoencoder

The key insight of Alternating Direction Method of Multipliers (ADMM) [8] is to divide the objective we wish to minimize into two (or more) pieces. One then optimizes with respect to one of the pieces while keeping the other pieces fixed. Accordingly, as inspired by the RPCA literature [15, 48] we split the objective in (12), (15), (7) and (16) into a first piece which depends on L_D and is independent of S and a second piece which depends on S and is independent of L_D. In effect, we use a back-propagation method to train a (sparse) autoencoder to minimize $\|L_D - D(E(L_D))\|_2 (+\beta \|E(X)\|_{2,1})$ with S fixed, a proximal gradient to minimize

[1] Not to be confused with E. W. Dijkstra's algorithm for shortest paths in graphs

the penalty term $||S||_1$ or $||S||_{2,1}$ with L_D fixed, and after each minimization we use element-wise projections $S = X - L_D$ and $L_D = X - S$ to enforce the constraints. The following is our proposed optimizing method:

Input: $X \in R^{N \times n}$, $\lambda \in R$ and $\beta \in R$ is optional for $\ell_{2,1}$ sparse autoencoder.
Initialize $L_D \in R^{N \times n}, S \in R^{N \times n}$ to be zero matrices, $LS = X$, and a (sparse) autoencoder $D(E(\cdot))$ with randomly initialized parameters.
while(True):

1. Remove S from X, using the remainder to train the autoencoder. (In the first iteration, since S is the zero matrix, the autoencoder $D(E(\cdot))$ is given the input X):

$L_D = X - S$

2. Minimize the (sparse) autoencoder using back-propagation.

3. Set L_D to be the reconstruction from the trained (sparse) autoencoder:

L_D = $D(E(L_D))$

4. Set S to be the difference between X and L_D:

$S = X - L_D$

5. Optimize S using a proximal operator:

$S = prox_{\lambda, l_{2,1}}(S)$

or:

$S = prox_{\lambda, l_1}(S)$

6.1 Check the convergence condition that L_D and S are close to the input X thereby satisfying the constraint:

$c_1 = ||X - L_D - S||_2 \ / \ ||X||_2$

6.2 Check the convergence condition that L_D and S have converged to a fixed point:

$c_2 = ||LS - L_D - S||_2 \ / \ ||X||_2$

if $c_1 < \varepsilon$ or $c_2 < \varepsilon$:

break

7. Update LS for convergence checking in the next iteration:

$LS = L_D + S$

Return L_D and S

Optimizing a (sparse) autoencoder using back-propagation is an off-the-shelf technique and well described in the literature [24, 25, 34]. In our experiment, we implement the (sparse) autoencoder part using Tensorflow [1] which tackles error back-propagation automatically. In particular, we describe the details of the "initialize a (sparse) autoencoder $D(E(\cdot))$ with randomly initialized parameters" step in the previous algorithm as follows:

Input: $X \in R^{N \times n}$, hList is a list which contains hidden layer sizes and $\beta \in R$ is optional for sparse autoencoder.

1.Initialize parameters for autoencoder W_{list} and b_{list}:

for i in range(len(hList)):

Initialize $W \in R^{hList[i] \times hList[i+1]}$, $b \in R^{hList[i+1]}$

Add W, b into W_{list}, b_{enlist}

Initialize b_{delist} with reversed order of b_{enlist}

2.Initialize a placeholder for Input X:

input_x with shape (batch_size, n)

3.Build computation graph:

last_layer = input_x

3.1 Encoding graph:

for W, b in (W_{list}, b_{enlist}):

hidden = sigmoid(last_layer $\cdot$ W + bias)

last_layer = hidden

hidden_layer = hidden

3.2 Decoding graph:

for W, b in reversed(W_{list}), b_{delist}:

hidden = sigmoid(last_layer $\cdot$ Transpose(W) + bias)

last_layer = hidden

recon = last_layer

4.Build the cost function:

if β is given (sparse autoencoder):

cost = $\|input\_x - recon\|_2 + \beta\|hidden\_layer\|_{2,1}$

else (normal autoencoder):

cost = $\|input\_x - recon\|_2$

Step 2 in our optimizing method, "Minimize the (sparse) autoencoder using back-propagation", is tackled by Tensorflow automatically. After training, reconstruction can be computed through the Tensorflow "recon" function.

11.2 Proximal Method for ℓ_1 and $\ell_{2,1}$ Norm

The ℓ_1 norm can be optimized efficiently through the use of a proximal method [7, 8] such as

$$prox_{\lambda,l_1}(x_i) = \begin{cases} x_i - \lambda, & x_i > \lambda \\ x_i + \lambda, & x_i < -\lambda \\ 0, & x_i \in [-\lambda, \lambda]. \end{cases} \tag{23}$$

Such a function is known as a *shrinkage* operator and is quite common in ℓ_1 optimization problems. For additional details see [8]. The following pseudo-code provides an implementation of proximal operator $prox_{\lambda,l_1}(S)$:

Input: $S \in R^{m \times n}$, λ

For i in 1 to $m \times n$:

    if $S[i] > \lambda$:

        $S[i] = S[i] - \lambda$

    if $S[i] < -\lambda$:

        $S[i] = S[i] + \lambda$

    if $-\lambda <= S[i] <= \lambda$:

        $S[i] = 0$

Return S

Similar to (23) the $\ell_{2,1}$ norm minimization problem can also be phrased as a proximal problem, but in a slightly more complicated form. In particular, the proximal operator for the $\ell_{2,1}$ norm is a block-wise soft-thresholding function [7, 6, 45]

$$(prox_{\lambda,l_{2,1}}(x))^j = \begin{cases} x_g^j - \lambda \frac{x_g^j}{||x_g||_2}, & ||x_g||_2 > \lambda \\ 0, & ||x_g||_2 <= \lambda, \end{cases} \tag{24}$$

where the g is a group index and j is a within-group index. This optimization

combines elements x_j into blocks and thus the element-wise sparsity from (23) becomes block-wise sparsity. For additional details see [8]. The following pseudo-code provides an implementation of proximal operator $prox_{\lambda,l_{2,1}}(S)$:

Input: $S \in R^{m \times n}$, λ

For j in 1 to n:

$e_j = (\sum_{i=1}^{m} |S[i,j]|^2)^{1/2}$

if $e_j > \lambda$:

For i in 1 to m:

$S[i,j] = S[i,j] - \lambda \frac{S[i,j]}{e_j}$

if $e_j <= \lambda$:

For i in 1 to m:

$S[i,j] = 0$

Return S

11.3 Experimental Evaluation

Since the objective function of both RDA (12) and RSA (16) are not convex, the convergence of our model to a global minimum is non-trivial to guarantee. However, in our experiments, we have empirically observed the convergence rates of our algorithm, and in this section, we provide some of our observations.

Back-propagation is the commonest training algorithm for deep learning and many packages, e.g. Tensorflow [1], implement automatic back-propagation of errors. However, such a method is not ideal for the optimization problem we are interests in. Figure 17 shows convergence records for selected values of λ. As one can see, training algorithm inside the Tensorflow cannot achieve quick and flat

converging plots for RDAs.

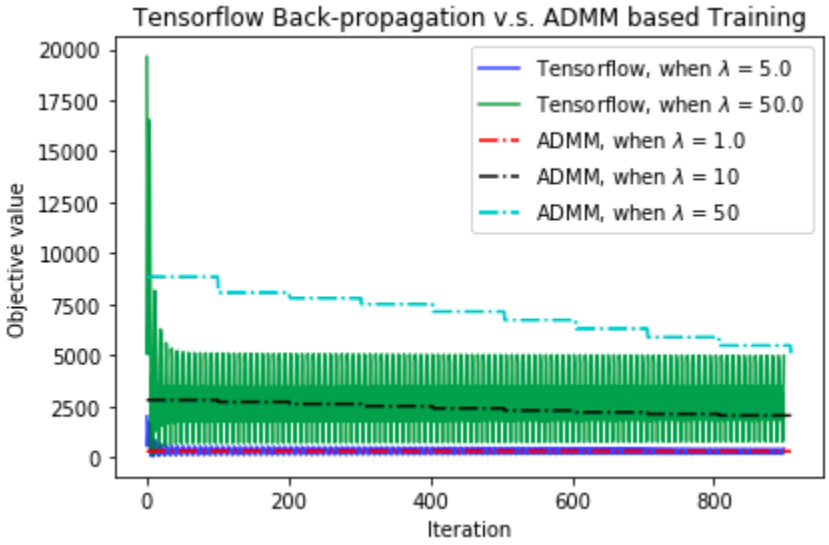

Figure 17: This figure shows a comparison of the convergence of RDAs directly trained by Tensorflow with our proposed training algorithm. The Tensoflow training are indicated by the solid lines and one can find that the objective descends quickly in the first iteration, but keeps bouncing even after 400 iterations with different λ. Convergence is hard to achieve for RDAs. However, for our training algorithm which is indicated by dashed lines, convergences seems slower but there is no bouncing trends in any of dashed lines. By comparing these two, we claim our proposed training algorithm is more stable.

Our proposed training algorithm, following the idea of an ADMM, minimizes one part of the objective with the other parts fixed. In particular, with S fixed, we use 30 instances as a training batch and 5 epochs to train the autoencoder part $|L_D - D_\theta(E_\theta(L_D))\|_2$. With L fixed, solving the proximal problem for S is deterministic and only takes one step. Accordingly, in Figure 18, one can see a "stair case" pattern where 5 epochs of minimization for the autoencoder is followed by

a single solve of the proximal problem for S. As one can see, the largest decrease in objective function is achieved by each proximal solve. λ is also essential to the convergence of the problem. Accordingly, Figure 18 shows convergence histories for several values of λ. As one can see, our proposed ADMM algorithm converges quickly in many cases of interest.

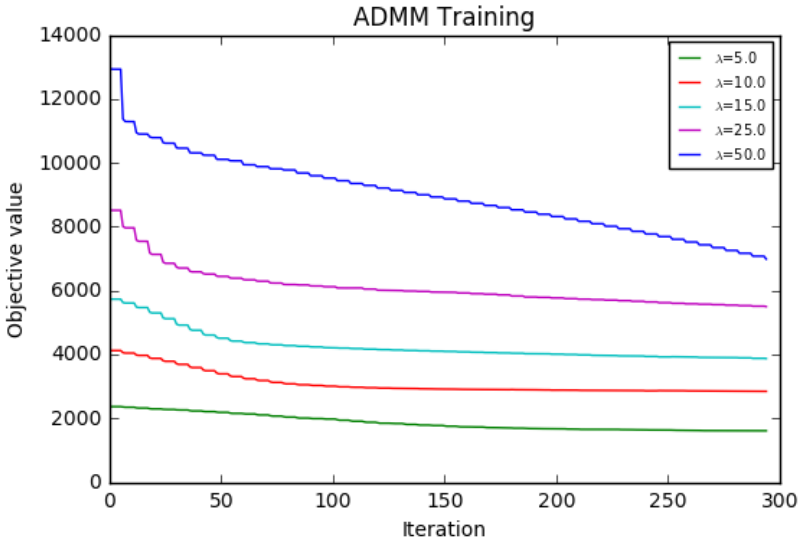

Figure 18: This figure shows the convergence of our optimization algorithm for a RDA. The objective descends quickly in the first 50 iterations and archives convergences after 200 iterations when λ is small. Convergence is slower for large λ values but still reasonable for our problems of interest.

11.4 Section Summary

We proposed a training algorithm for the optimization of RDA and RSA which is similar to traditional Robust PCA but with more complicated objective functions that is not convex and not differentiable. This proposed algorithm combines ideas from proximal methods [8], backpropagation [56], and the Alternating Direction of

Method of Multipliers (ADMM) [7]. We demonstrated experimental effectiveness and convergence of our algorithm. The theoretical proof of convergence is one part of our future work.

Part III

Applications to Cyber Security

12 Overview

Distributed Denial of Service (DDoS) attacks are a constant danger to today's Internet [2, 12, 13, 14], and not all such attacks can be detected using signature based approaches [14]. Accordingly, anomaly-based machine learning and data mining methods are appealing to the defenders of networks because of their ability to detect zero-day attacks where no templates are available. As described in Part I, our development of RPCA and RDA allows detecting anomalies without noise-free data as references. Herein, we detail our work that applies RPCA and RDA on cyber data and demonstrate their superior accuracy when compared to baseline methods. Beyond that, we modify the standard RPCA to analyze attacks in a vast and variety streaming environments. Our experiments qualitatively and quantitatively demonstrate the performance of our proposed algorithms using a collection of data arising from Raytheon BBN's high-fidelity simulated network test-bed on a variety of real world DDoS attack vectors.

13 High-fidelity Simulated Data

Sand-box data is commonly used in cyber attack analysis [11, 14]. As a consequence, the quality of simulated data plays an important role to the success of analysis. This section provides a short overview of the data generation work and also this section is a special thanks to the data provider, Raytheon BBN Technology.

The techniques is evaluated using data collected from the ARMED project. ARMED is a DARPA funded project that is developing technologies to defend network enclaves hosting services provided to clients outside the enclave against sophisticated distributed denial of service attacks. ARMED uses instrumented network stacks, called ARMED Network Actors or ANAs for short, placed in the network enclave that form an overlay that all traffic to the defended services must flow through. The ANAs act as transparent relays for the clients– i.e., from their perspective they still interact with the intended endpoint, whereas protocol specific instrumentation at the ANAs may terminate TCP or HTTP connections, collect data that is subjected to analysis such as RPCA, and may inject responses without having to connect to the intended endpoint. The presence of ANAs incur overhead, and depend on how many protocol-specific instrumentation points the traffic has to cross before hitting the intended service endpoint, in our evaluation so far, we have observed the overhead to up to 30% (which corresponds to a net increase in RTT of 75ms in the test-bed).

The test-bed, emulating a small services enclave containing a DNS and an HTTP-based web service, is defended by a 2-tier configuration of ARMED ANAs: the first tier is responsible for handling the TCP protocol, and the second tier is responsible for handling the HTTP protocol.

To increase the realism of our evaluation, a sophisticated mechanism, called Yoshka [43], is applied for background traffic generation. Yoshka makes use of behavior trees for traffic generation. The behavior trees model user workflows/activities (e.g. downloading a file, sending an email). Thus, the execution of an activity depends on the success or failure or previous activities. The list of tasks per user is executed on a loop following a non-deterministic approach using a uniform distri-

bution with a random interval between 1 and 10 seconds. Yoshka supports a set of protocols such as HTTP, FTP, GIT, IMAP, SMTP, and SQL. In the current ARMED test-bed, Yoshka typically emulates 140 users . In the results presented in this paper, the user model used for background traffic contained only FTP and HTTP interactions.

14 Related Work

14.1 Novel DDoS Attack

There exists a large extension literature which keeps eyes on the increasing threaten on the cyber world. Over the last decades, Internet has increasingly become the critical communication medium of the world. Enrico Cambiaso etc. [12] present a novel threat, SlowDroid, on Android system. SlowDroid implements a Denial of Service attack and it is particularly suitable to a mobile execution since it makes use of low amounts of computational and bandwidth resources. Similarly, another novel threat called SlowComm shows its success on leading a DoS on a targeted system using a small amount of attack bandwidth [13]. These attacks share the same mechanism with the Slow Read attacks that we are working on.

Beyond direct attacks, malicious users neutralize security restriction through protocol encapsulation, tunneling peer-to-peer, chat, or HTTP packets into allowed protocols such as DNS or HTTP [2]. An innovative profiling system is proposed in [2] for DNS tunnels which is based on Principal Component Analysis and Mutual Information. These techniques share the similar idea with Robust Principal Component Analysis that Slow-communication attack, including Slowread, SlowLoris, SlowDroid, and SlowComm, significantly increase similarity between data which

is a key feature to protect cyber systems.

14.2 Anomaly Detection with Cyber Data

There is a large amount of research works trying to apply machine learning and data mining techniques on analyzing cyber security data. Most of them falls into two main types of cyber analytics in support of analyzing: signature-based and anomaly-based [11]. Here, we mention two example of research works in the literature: Hu et al. [28] used the robust support vector machine (RSVM), a variation of the SVM where the discriminating hyperplane is averaged to be smoother and the regularization parameter is automatically determined, as the anomaly classifier in their study. The application of Random Forests to anomaly detection is described by Zhang et al. [67], where an anomaly (outlier) detector was employed to feed a second threat classifier.

However, these methods are supervised methods which require an access to the pre-defined label for the training process which have difficulties on detecting new attacks, which is well-known as zero-day attacks [11]. Anomaly-based techniques model the normal network and system behavior, and identify anomalies as deviations from normal behavior. Our models falls into this category which are appealing to us because of their ability to detect zero-day attacks [11].

15 Robust PCA for Anomaly Detection in Cyber Data

As we introduce in Section 3.1, RPCA is classically used to construct low dimensional *linear* features by filtering out outlying measurements [48]. In this section, we demonstrate how this technique can be used to analyze the dimension and

detect anomalies in data arising from Raytheon BBN's high fidelity simulation infrastructure and make use of RPCA derived features to identify different DDoS attacks. From our results, which are also our main contribution in this work, we find that DDoS attacks has closely correlated to the dimension changing and the sparse, high-dimensional values' variation when networks are actually under the DDoS attack.

In general, as we introduce in Section 3, Robust PCA allows splitting X into two parts, namely L and S. In this section, we detect cyber attacks through two important features that are generated by such a splitting of Robust PCA. In particular, we compute the dimension of L to measure the similarities between instances and features and use the summation of columns in S to measure the importance of anomalous features.

Robust PCA provides the careful teasing apart of sparse anomalies so that the remaining low dimensional approximation is faithful to the noise-free low-dimensional subspace describing the bulk of the raw data [48]. In our analysis, our input data is X, and we describe the features derived from Raytheon BBN's high performance simulator in Section 15.1.

15.1 Application In Cyber Security

Our data is generated from a simulated network which topology is shown in Figure 19. Normal users and potential attackers connect to an HTTP server and a DNS server through a sensor network. ARMED Network Actors (ANA) sensors record the network activities and each ANA sensor monitors different subnets. As an example, one of our data has over 1,500,000 instances and each instance has 13 raw features. The other files provided to WPI by Raytheon BBN following a sim-

ilar convention. These raw features include *timestamp, window, offset, destination port, source port, destination address, source address, sequence number, RST flag, FIN flag, ACK flag, SYN flag, and state.* These raw features are recorded according to monitoring data transferred by Transmission Control Protocol (TCP). Herein, we list feature meaning in detail [16]:

- *Timestamp* indicates the time of data being recorded.
- *Window* specifies the size of receive window.
- *Offset* specifies the size of the TCP header in 32-bit words.
- *Destination port* identifies the receiving port.
- *Source port* identifies the sending port.
- *Destination address* identifies the receiving IP address.
- *Source address* identifies the sending IP address.
- *Sequence number* If the SYN flag is set, then this is the initial sequence number. If the SYN flag is clear, then this is the accumulated sequence number of the first data byte of this segment for the current session.
- *RST flag* presents the willing of resetting the connection.
- *FIN flag* indicates the last packet from sender.
- *ACK flag* indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client will have this flag set.
- *SYN flag* is used to synchronize sequence numbers with server. Only the first packet sent from each end will have this flag set.

- *State* indicates the states of current connection.

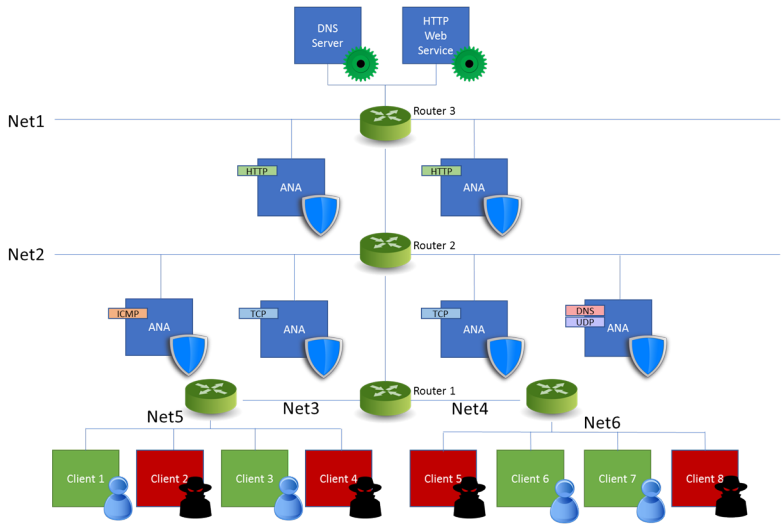

Figure 19: This figure shows the basic setup of the network that generates our data. There are normal clients, potential attackers, and DNS and HTTP servers in the network. The network is monitored by ANA sensors. Each ANA sensor works individually and monitors different subnets and protocols at different network layers.

Denial-of-service (DoS) attacks are characterized by an explicit attempt by attackers to prevent legitimate use of a service. There are two general forms of DoS attacks: those that crash services and those that flood services [66]. The most serious attacks are distributed.

There are two types of distributed attacks in our data: "slow read" attack and "SYN-flood". A slow read attack sends legitimate application layer requests, but reads responses very slowly, thus trying to exhaust the server's connection pool. A SYN flood occurs when a host sends a flood of TCP/SYN packets. Each of these packets are handled like a connection request, causing the server to spawn a half-open connection and waiting for a packet in response from the sender address.

We clean the raw data to satisfy the assumptions of our Robust PCA model. Data cleaning consists of three steps:

1. Feature Selection: We only use the timestamp feature to slice the data into windows. We drop this feature since it is useless for Robust PCA analysis since it merely indexes the other features. We also delete the *window* and *sequence number* features since they start with random numbers and are useless for measuring distances and similarities (though they are quite useful for eventual integration of the Robust PCA detected anomalies into the larger ARMED systems).

2. Encoding IP and Port Number: Our network is based on IPv4 protocol. Both source and destination are IPv4 addresses which consists of 4 bytes. However, IPv4 addresses contain more information than just merely distinguishing unique addresses. The IPv4 address includes hierarchical information in its format: the leftmost bytes designate which top-level domain a particular address belongs to, and the rightmost bytes designate subnets within that top-level domain. The precise number of bytes which are used for the top-level domain and the subnets a function of the "class" of the network. To preserve this hierarchical information, we divide the IP feature into four features according to its byte and treat each byte as a categorical feature. The first three features categorize its hierarchical subnet and the last feature distinguishes different terminals in the subnet.

   The *port number* feature is categorical and it ranges from 0 to 65535. Some ports are monopolized by widely used protocols like 80 for HTTP and 443 for DNS while some ports are dynamic and randomly picked when needed. Giving each port number a single feature encoding is verbose and unnecessary. We want to distinguish port numbers according to their frequency. Thus, we encode the port number into 3 categories: "common", "registered",

and "dynamic".

3. OneHot Encoding and Z-scoring Data: Our features are categorical represented as integers but ordered integer numbers are misleading when considering distances between categories, since a naive encoding would force the distances between any two different categories to be equal. We use One-Hot encoding which transfers a single categorical feature to a group of features among its legal values. There is only those with a single "1" in the group which represent current choice and all the others are "0". The main advantage of One-Hot encoded data is that it gives comparable distances across different categories.

15.2 Experimental Evaluation

After cleaning, we divide instances into windows according to their timestamp. Each window contains 8000 instances. We apply Robust PCA to each window separately and demonstrate the dimension and the anomalies for each window and check how they vary along with time. *Both dimension of L and anomalies in S can help us identify certain types of attacks* and we give details in Figures 20 and 21. To help intuitively explain how the dimension and anomalies of data associate with attack traffic we provide more detailed visualizations on some time windows.

15.2.1 Dimension and Anomalies

The dimension of the traffic data is an important indicator of the data's structure and it measures similarity of network traffic. Network traffic of normal users are expected to be at a relatively stable level since normal users are neither extremely akin or extremely alien. Dramatic increments and decrements of dimensions indi-

cates abnormal network traffic. In particular, significant low-dimensional measurements indicate broad swaths of network traffic are identical and therefore probably generated by bots. On the other hand, a sharp increment in the anomaly part indicates some unexpected and unseen traffic which also could be an attack. Figure 20 shows the dimension of L changing with the time windows. We can tell that during the "slow read" attack, the dimension of network traffic data drops which demonstrates that the "slow read" attack makes the network traffic more similar with each other (and likely artificially generated).

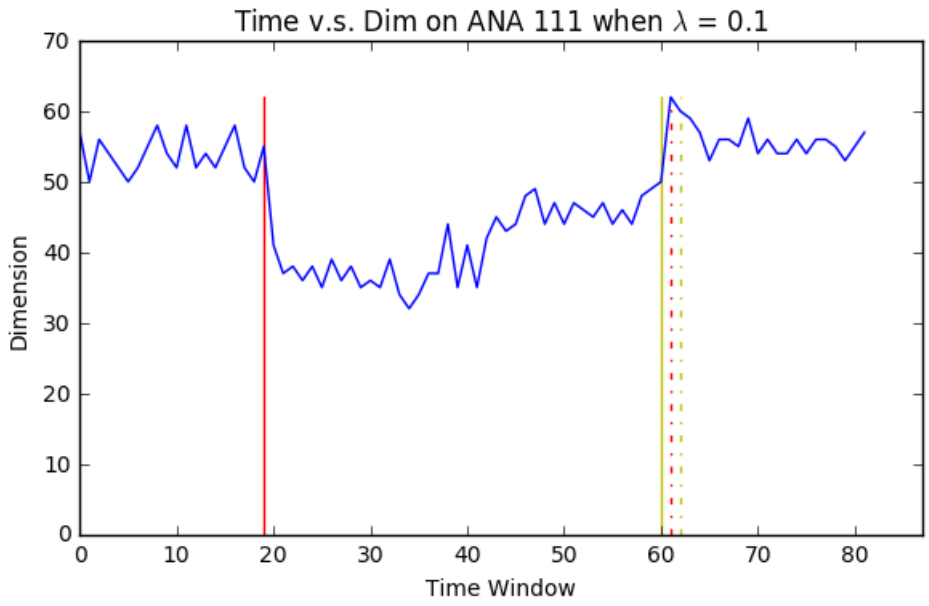

Figure 20: This figure shows the dimensions of data changing across time windows. The solid red line indicates the time of a "slow read" attack starting and the dashed red line indicates the time where it stops. The solid yellow line indicates the time of a "SYN-flood" attack starting and the dashed yellow line indicates the time where it stops. One can find that during the "slow read" attack, the dimension of the data is lower than when no attack occurs. Data measured during the "slow read" attack is much lower dimensional than others time windows which indicates such measurements are not likely generated by normal users. Such low dimensional behavior of a "slow read" attack is a primary result of our work.

Figure 21 shows the sum of absolute values of anomalies for each column in

S changing with the time window. In Robust PCA, the S part is considered as anomalous. We propose to use the absolute value of sums of columns of S to measure an abnormality level. Also, instead of treating S as one piece, we provide a detailed visualization of the abnormality level for different groups of features (such as port numbers). In Figure 21, the distance between lines demonstrates the abnormality of features. One can tell that abnormality changes with the time window and gets a sharp upper increment at the time of the yellow line indicating a "syn-flood" attack happening. The sharp increment in S implies there are huge number of unexpected source IP, port numbers and flags transferring through the network at the time of the "syn-flood" attack.

15.2.2 Network Connections under Different Attacks

To understand the network traffic in different environments, we visualize different examples of network features in which each node is an individual host and each edge is a connection between hosts. Every black box on the edges is the direction of an arrow for each connection that starts from its source host to its destination. Figure 22 shows an example of the normal traffic on time window 4.

Figure 22 shows an example of traffic under "slow read" attack in time window 33. Compared with the normal traffic in Figure 22, the "slow read" traffic is more similar which match our previous analysis that "slow read" reduces the dimensionality of the measured data.

Figure 24 shows an example of traffic under "syn-flood" attack in time window 61. One can find that there are many unexpected hosts connecting the server on the "syn-flood" attacks which increase the number of anomalies in the data. This phenomenon also matches our previous analysis in Figure 21.

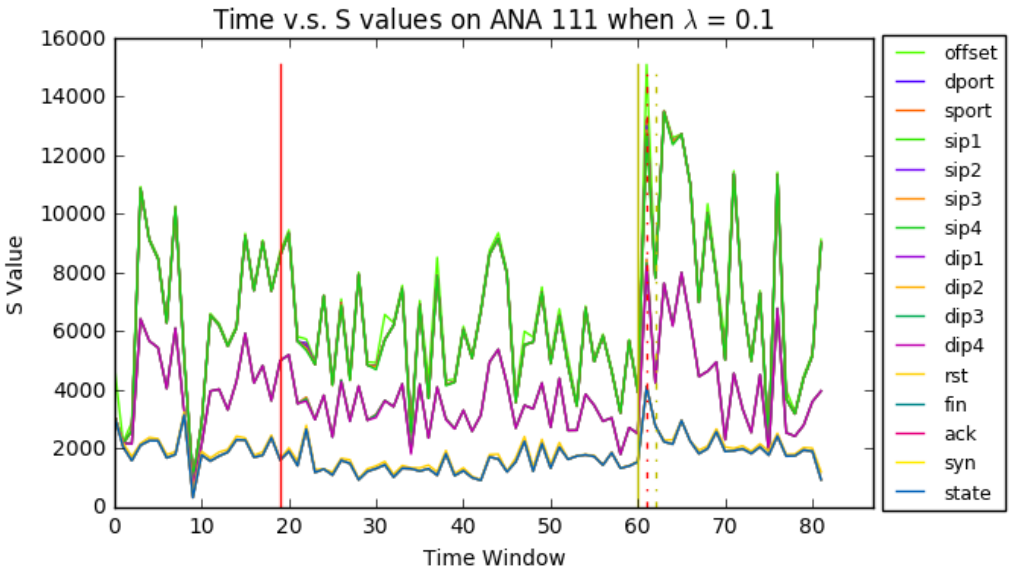

Figure 21: This figure shows the anomaly part S changing with the time window. In particular, each line shows a collection of related feature values in S changing with the time window. Like Figure 20, red and yellow lines mark the starting and ending times of two attacks. One can find that there is a spike during the "syn-flood" attack. This spike indicates that the "syn-flood" attack consists of large number of connections with unexpected IP, strange port number, and unanticipated state flags to the server. These unpredictable connections are high-dimensional and thus distinguished as anomalies by Robust PCA. Such a surge of anomalies under the "SYN-flood" attack is another primary discovery of our work.

15.2.3 Second Order Analysis

After the first order analysis of raw data, we begin a second order analysis where we compute the linear correlations between features for each time window and apply Robust PCA on correlation matrices. Correlations represent the linear predictability between features and the analysis of correlation matrices allows one to distinguish when the predictability between features changes.

As described in Section 15.1, we have 86 features, and the correlation matrix is 86×86 at each time window. The entry at i-th row and j-th column in a correlation matrix indicates the linear predictability between the i-th feature and j-th feature

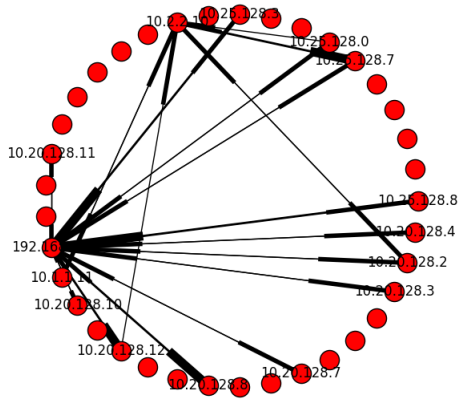

Figure 22: This figure shows an example of the network connections in the normal traffic on time window 4. One can find that the normal traffic are different but roughly contains two types of business traffic. There are many nodes connecting to the 10.2.2.10 and the 192.168.2.4 which are the two HTTP servers.

in the original data. Robust PCA splits correlation matrices into L and S, and S therefore contains anomalous correlations between features.

Comparing Figure 21 and Figure 25, we find a surprising but reasonable phenomenon that the "slow read" attack does not match any pattern in S in the *original data*, shown in Figure 21, but it does match a significant increment in S of *correlation matrix*, shown in Figure 25.

15.3 Section Summary

In this work, we contribute our major discovery that *Both the dimension and the anomaly level of data are important features for us to identify DDoS attacks*, since normal users are neither extremely akin or extremely alien. In our Robust PCA analysis, we propose to use the dimension of L as an indicator of the major part of data and the absolute value of sum of S to measure the anomaly level of data. We also provide intuitive topology plots of networks which match the results of both the dimension and anomaly analysis. Last, we use second order analysis of Robust

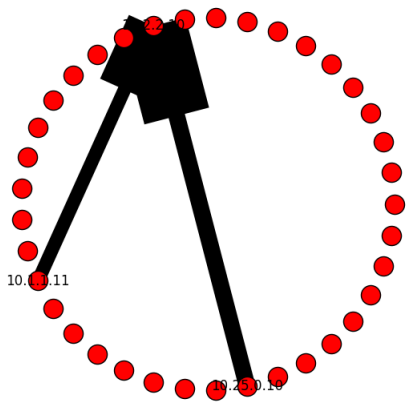

Figure 23: This figure shows an example of the network topology under "slow read" attack on time window 33. There are only two hosts connecting with an HTTP server 10.2.2.10. Both of them hold a large amount of connections with the server. These connections are similar, and thus data are low dimensional. We can find their low dimensional behavior in Figure 20.

PCA which enhances our conclusion that some types of DDoS attacks increase the correlation between features.

The RPCA-based analysis of network data described in this thesis is a key component of the ARMED technology suite developed under the DARPA XD3 program. ARMED (Adaptive Resource Management Enabling Deception) technology adds an in-network maneuvering capability to service enclaves– network enclaves hosting services that are consumed by clients that reside outside of the service enclave. The primary goal of the ARMED maneuvers is to defended the services hosted in the service enclave from sophisticated denial of service attacks from outside clients. ARMED technology, including the RPCA analysis component has been demonstrated and evaluated against a number of real and synthetic attacks including the segmentsmash attack and the SRI DDoS Laboratory tool. ARMED is currently being evaluated for transition to defend a tactical situational awareness server that is widely used by various civilian and military agencies.

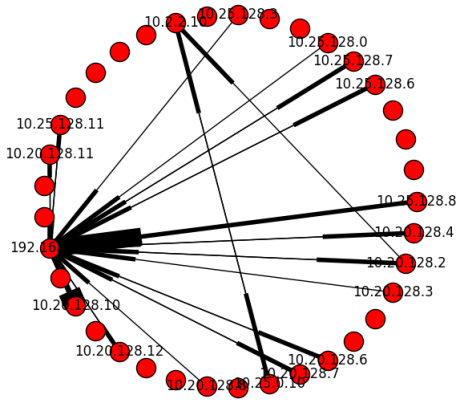

Figure 24: This figure shows an example of the network topology under "syn-flood" attack on time window 61. One can find that many nodes connect to the server 192.168.2.4. However, many of them are unexpected and differ with the topology of normal traffic. As a consequence, Robust PCA filters out many of values into S part. Figure 21 shows that there is a spike at this time window.

16 RPCA and RDA for Semi-supervised Learning of Attacks

16.1 Semi-supervised Learning

We introduce algorithms of applying semi-supervised learning on cyber data. These algorithms first implement a dimension reduction technique: RPCA or RDA, then train a Random Forest on the lower dimensional data and predict whether or not an entry is an attack.

We begin with the setup of our data. Our data is in a matrix $X \in \mathbb{R}^{N \times m}$, where N is the number of entries and m is the number of features from a buildt-in sensor that monitors a certain aspect of the network. This large matrix is then split up into new matrices, or slices, by every $8\,000$ rows. Thus we have a new matrix $Y \in \mathbb{R}^{a \times b \times m}$, where a is the number of slices, b is the number of entries in each slice ($8\,000$), and m is still the number of features. In our data set, *we know that attacks*

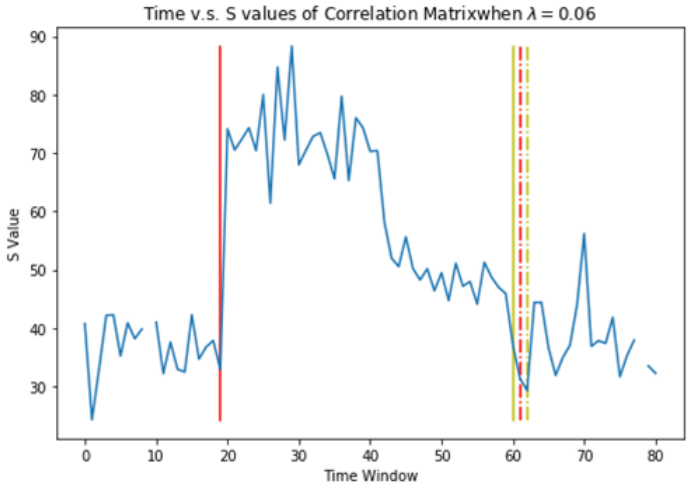

Figure 25: This figure shows a summation of the anomalous part of S in correlation matrices changing with the time window. Like Figure 20, red and yellow lines mark starting and stopping time of two attacks. The solid red line indicates the time of a "slow read" attack starting. One can find that there is a sharp increment when the "slow read" attack starts at time 19. This increment indicates that the "slow read" attack significantly increases the correlation between features.

only occur on slices 24-40 and therefore we will focus on those. Slices 0-23 do not contain any attacks. This can be seen in Figure 26.

16.1.1 Semi Robust Principal Component Analysis

For this method, we combine Algorithm 1 and 2. The first algorithm takes in Y and Lam, where $Y \in \mathbb{R}^{a \times b \times m}$ as stated in Section 16.1 and Lam is an array of the possible λ values for RPCA. As mentioned in Section 3.1, λ is the coupling constant which tunes how sparse the S matrix is. Therefore, by training a variety of possible values for λ, and cross-validating the results, the most accurate value of λ can be selected for the final model.

First, in our experiments, the attacks only happened in slices 23-40 in our dataset where we build a list that contains the labels, as shown in Algorithm 1. Then the algorithm iterates through each possible λ value. The very first slice of clean data

is used to fit RPCA to, and is thus decomposed into a low rank L and a sparse S matrix. Then, for each slice that contains attacks, the slice is projected down onto the same lower dimensional plane as the initial slice of clean data. This produces the low rank L matrix for each slice and S is obtained by subtracting L from the slice. Each sparse matrix of outliers S is saved for later analysis.

Once each S matrix is obtained for each slice and each possible λ, the next step is training the classification algorithm. Algorithm 2 shows the training of the classification algorithm. This algorithm takes in comb and S. In this case, comb is an array of the different λ values and $S \in \mathbb{R}^{c \times a \times b \times d}$, where c is the number of λ values, a is the number of slices, b is the size of each slice, and d is the dimension that RPCA projected the data down to (i.e., S is the results saved from Algorithm 1. Arrays for False Positive Rate (FPR) and False Negative Rate (FNR) are initialized as empty arrays. Then for every λ a Random Forest Classifier (clf) is fit to the first slice of attack data for which that attack was occurring the entire time (this is the second slice of attack data) for that given λ. Then for each of the remaining slices related to that λ, the clf predicts which entries are attacks. The FPR and FNR arrays are filled with the False Positive Rates and False Negative Rates respectively for each slice and λ combination.

Algorithm 1 Semi-Supervised RPCA

Input: $Y \in \mathbb{R}^{a \times b \times m}$, Label is a boolean array indicates whether a slice contains attacks, and Lam is an array.

mixture_slices = Y[Label=='att']

normal_slices = Y[Label=='normal_only']

for all lam $\in$ Lam:

$L, S =$ rpca(normal_slices, lam)

for slice $\in$ mixture_slices:

projectedSlice $=$ project(slice, L)

S $=$ slice $-$ projectedSlice

save S

Algorithm 2 Random Forest Classification

Input: $S \in \mathbb{R}^{c \times a \times b \times m}$, Label is a boolean array indicates whether a slice contains attacks, Lam is an array contains of different λ, FPR and FNR are function calls that compute false positive rates and false negative rates respectively.

mixture_slices $=$ Y[Label==’att’]

for slice $\in$ mixture_slices:

Initialize: FPR_list and FNR_list

for lam $\in$ Lam:

classifier $=$ RandomForest.fit(slice[lam,1])

prediction $=$ classifier.predict(slice)

FPR_list.append(FPR(prediction))

FNR_list.append(FNR(prediction))

for lam $\in$ Lam:

res $=$ FPR_list[lam] * FNR_list[lam]

bestLam $=$ argmin(res)

Plot FPR_list[lam] and FNR_list[lam]

Next, we needed to select the best combination of parameters that are being trained (λ for RPCA). For each λ value, the False Positive Rate for Slice 2 (third slice containing attacks) was multiplied with the False Negative Rate for Slice 2 (third slice containing attacks). The λ value with the lowest result from this multiplication will be selected as the combination. This allows the result to depend more on the value which varies the most. For instance, in our dataset, the FPR are all very similar between different combinations while the FNR can very rather significantly, with our proposed approach having the best performance in all cases.

16.1.2 Semi Robust Deep Autoencoder

For this method, we combined Algorithm 2 and 3. This method takes in four parameters: Y, Lam, Label and Inner. Y and Lam are the same as explained in Section 16.1.1 and Inner is an array of possible inner layer dimensions. As mentioned in Section 5.1, the Robust Deep Autoencoder contains a low-dimensional hidden layer, and Inner contains possible dimensions of this hidden layer.

This algorithm begins with utilizing the labels that discovered as in Section 16.1.1 to select slices that contain attacks. Then the algorithm iterates through every combination of λ and hidden layer values. Then a RDA is trained with the λ and hidden layer dimensions combination. The RDA is then fit to the first slice of data provided, which does not contain any attacks. Once the S and L matrices are obtained, each slice of attacks is iterated through. Three matrices are obtained for each slice. First, the fully reconstructed L matrix is the $D_\theta(E_\theta(L_D))$ value in equation 12. The S matrix is directly from equation 12, and the hidden layer L matrix is $E_\theta(L_D)$. Finally, we save the S matrix by itself, and the S matrix appended with the reconstructed L matrix such that the returned matrix is of size $b \times 2m$, and S

matrix appended with the hidden layer L matrix such that the returned matrix is of size $b \times (m + hidden)$. Therefore for each of the possible λ and hidden layer combinations, we have 3 returned matrices for each slice.

Algorithm 3 Semi-Supervised Robust Auto-encoder

Input: Y, Lam is an array contains of different λ, Label is a boolean array indicates whether a slice contains attacks, and Inner is an array of possible inner layer dimensions.

```
mixture_slices = Y[Label=='att']
normal_slices = Y[Label=='normal_only']
for lam ∈ Lam:
    for hid ∈ Inner:
        rae = RDA.init(lam,hid)
        L,S = rae.fit(normal_slices)
        for slice ∈ mixture_slices:
            reconL = D_θ(E_θ(slice))
            S = slice - reconL
            hiddenL = E_θ(slice)
            save S+reconL, S+hiddenL, S
```

Each of these matrices forms a separate testing algorithm, thus Algorithm 2 is run 3 separate times for each output of Algorithm 3. The parameter comb now represents each λ and hidden layer dimension combination for which the data was found. We can now define three methods:

- S RAE: where the Random forest is trained on S,

- SL RAE: where the Random forest is trained on S and the reconstructed L, and
- SL hid RAE: where the Random forest is trained on S and the hidden layer that gives rise to the reconstructed L.

For each combination of λ and hidden layer dimension, the Random Forest is trained on slice 1 (the second slice with attacks in the dataset), and then predicts on all the remaining slices. The best combination of λ hidden layer dimension is selected from slice 2, as explained in section 16.1.1.

16.2 Experimental Evaluation

To determine the accuracy of each of our models at determining if an entry is an attack, we compared FPR and FNR. The FPR is the total number of false positives (falsely labeled attacks) predicted out of all positives (attacks) predicted. The FNR is the number of false negatives (falsely labeled non-attack data) out of all negatives (non-attacks) predicted. Each method produces a predicting label for each data entry and from this information, we compare the FPR and FNR of each of our methods. Thus, the lower the FPR and FNR of a method, the more accurate the method is at predicting attacks in our data. Additionally, we compare some baseline methods to the more complex methods developed in this paper to determine the improvement from industry standard methods.

16.2.1 Baseline Methods

We develop two baseline methods: Random Forest and Principal Component Analysis (PCA).

Random Forest (RF) performs no dimension reduction on the data and simply fits a Random Forest Classifier to the first slice of attack data and then uses that classifier to predict on all the other slices of attack data.

Principal Component Analysis (PCA), performs the basic form of linear dimension reduction before classifying the data. This method fits a PCA model to a slice of data without any attacks to find the lower dimension of that data set. More in depth information on PCA can be found in [29]. From there, each slice of data with attacks is then projected onto that same lower dimensional subspace. Once each of these lower dimensional subspaces of the original data are formed, a Random Forest Classifier is trained on an attack slice of data and then the classifier is used to predict the labelling of the rest of the attack slices.

16.2.2 Data Labeling

For the semi-supervised portion of our results, we needed to determine labels for each entry in the dataset. Throughout the given data, a type of denial of service attack occurs called slow-loris. For more information on slow-loris attacks reference [14]. This attack, in our data, is known to be initialized by one specific source IP address ('10.25.0.10') and is sent to another destination IP address ('10.2.2.10') through a given destination port (80). Additionally, we know the range of time during which the attack occurred. Therefore we can use these pieces of information and label attacks based on the entries source IP, destination IP, destination Port, and time. The overall dataset with attacks labeled in red can be seen in Figure 26.

To organize our data, we will divide our dataset up into slices. Every 8,000 entries are a slice. There are 17 slices which contain attacks within them. These

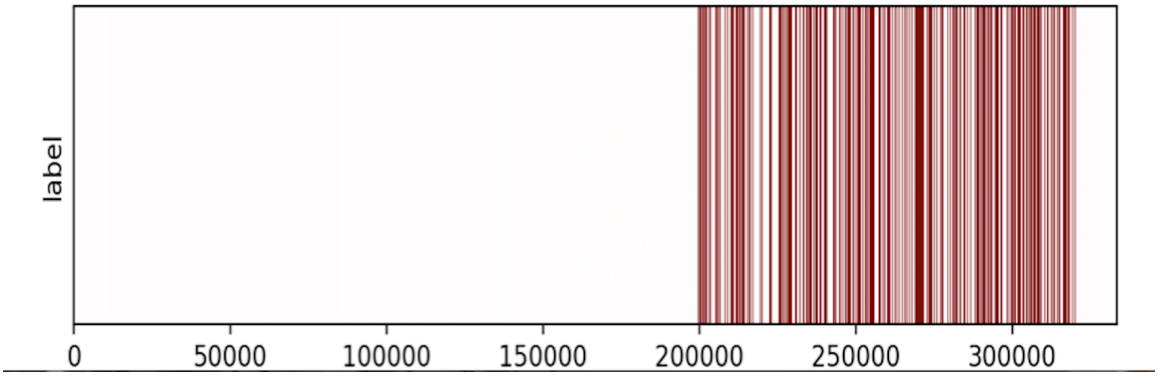

Figure 26: This figure shows the entries in the dataset which are labeled as attack entries. The first half of the dataset contains no attacks, while the second half is under attack. Thus, we hope to predict the attack entries in the second half of our dataset.

slices will be focused on in our later results.

Additionally, for our semi-supervised results, we needed to remove the features used in manually labeling attacks. Therefore, we removed all features which were created from IP addresses for this portion of our results. This resulted in reducing the 86 features down to only 33 different features. After deleting IP and port numbers, our algorithms only rely on the rest of features rather than the IP and port numbers which are used to create labels.

16.2.3 Semi-supervised learning results

For this section, we trained and tested six different models on 17 slices of data which included slow-loris attacks. The first model (RF) trained a Random Forest on the cleaned data set. The second model trained PCA on the dataset and trained a Random Forest on the linear projection of slice 1 onto a lower dimensional subspace. The third model (RPCA) was described in section 16.1.1, and the last three models: S RAE, SL RAE, and SL hid RAE are explained in 16.1.2. For the unique models to this paper, RPCA, S RAE, SL RAE, and SL hid RAE, we trained the di-

mension reduction model on the first clean slice of data in the dataset, trained the classification model on slice 1 of the data with attacks, and cross-validated the results on slice 2 of the data with attacks to determine the best values for λsl rae or λ and hidden layer dimension. We trained on slice 1 instead of slice 0 of the attack since the attack did not start until half way through slice 0. The results were more accurate with slice 1 since that slice was more similar to the others with attacks. Next, we aimed to make these methods plausible for real time analysis, so we cross-validated the results on slice 2 to select the optimal λ or λ/hidden layer dimension combination. Thus, once the first 2 slices are complete, the most accurate algorithm would have been selected and could be run on new data as it came into the network. In Figure 27 these methods were trained and cross-validated on Slices 1 and 2, and only the accuracy on the "testing" Slices 3 can be thought of as being representative of the true accuracy of the algorithms.

In addition in Figure 27, the FPR of methods are compared over different slices. Each method has a very similar and low FPR throughout each of the slices, except for the last slice. The last slice causes a variation of the FPRs, which could be due to the fact that the attack ends half way through the slice and therefore there are significantly less attacks in that slice. If the algorithm predicts about the same number of attacks as the other slices, then there would be more entries predicted as attacks than are truly attacks. Therefore, there would be more false positives, which is what is occurring in this slice.

Finally, in Figure 27, the different methods proposed in this paper show improvement from the baseline methods. The baseline methods continually have the highest false negative rates, reaching as high as .7 in slice 12. This shows that the methods are unable to mark the majority of true attacks as attacks. Since the objec-

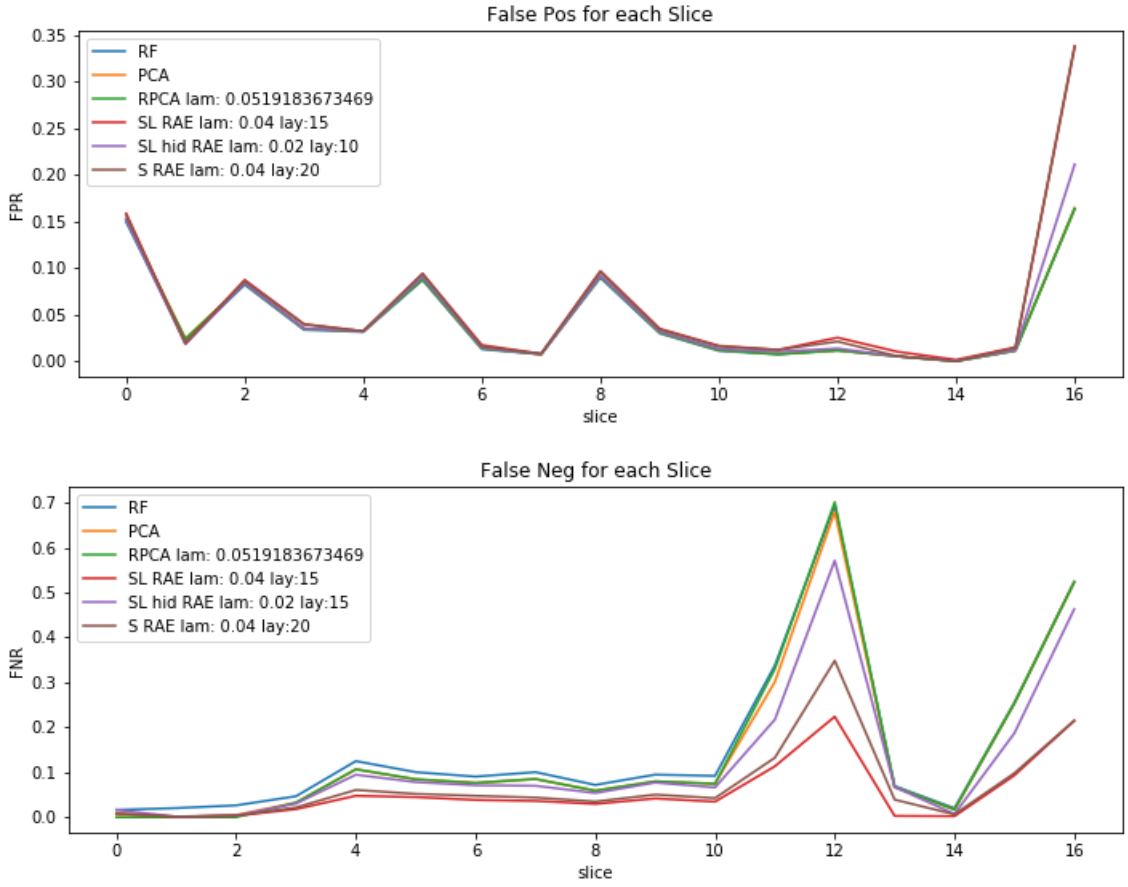

Figure 27: On the top, we show False Positive Rates for each of our semi-supervised methods. For the methods proposed in this paper, the λ and hidden layer dimension selected for the optimal model are stated. Each method contains very close results to one another and the FPR is fairly low for each slice except the last. On the bottom, we show False Negative Rates for each of our semi-supervised methods. Each of these methods were trained on beginning slices and then used to predict attacks on later slices. For the methods proposed in this paper, the λ and hidden layer dimension selected for the optimal model are stated. S and L Robust Auto-encoder methods clearly outperforms the other methods in every slice. *These False Negative Rates are much lower for our proposed methods than for the comparison methods.*

tive of anomaly detection is to not miss any attacks on the network, this accuracy is very poor. The method SL RAE, in which is S concatenated with the fully reconstructed L, has a lower FNR for each slice than any other method, reducing the false positive rate to 0.2 from 0.7!

16.3 Section Summary

In this section, we show how the unsupervised methods of RPCA and RDA, a method which combines the nonlinear dimension reduction of autoencoders with the anomaly detection characteristics of RPCA, can be combined with the supervised method of Random Forests to create a semi-supervised method for detecting anomalies. We compare these different methods against baseline methods as well as each other to demonstrate a significant improvement in the accuracy of predicting attacks. We determine that our RDA methods obtain significantly higher accuracies, when predicting what traffic is attacks, than the other methods.

17 Streaming Anomaly Detection via Robust PCA

Streaming anomaly detection is the ability to identify abnormal instances while additional data is generated over time. Our focuses in this section adapts out static anomaly detecting technique, Robust PCA, to streaming environments. We view such streamed Robust PCA process as performing actions of Robust PCA on data incrementally as it arrives rather than collecting windows of data for batch processing. Our initial problem comes from our data which is $X \in N \times k$, where N is the number of instances and k is the number of sensors. It is very common in real-world cyber security problems that $N >> k$, since k is fixed after sensor networks

built up, while N can be arbitrary large for it continuously increases with new instances generated. In particular, we are considering problems with a tremendously large N that fitting data into main memory is impossible. Such a memory limitation obliges us to drop long history of data and only maintain pivotal information in the main memory, and keep update it upon new data coming.

17.1 New Challenges for Second Order Analysis

The second order analysis is promising to relieve memory usage in two ways. First, it is able to reach required accuracy of analysis that second order analysis captures both types of attacks. Also, second order analysis reduces the size of data from a shape of $N \times k$ to a size of $k \times k$, where $N >> k$. It is therefore feasible to solve the Robust PCA optimization on second order, covariance matrices with the size of $k \times k$ than the first order, raw matrices with the sizes of $N \times k$. Even through developing fast and stable second order analysis techniques can be considerably advantageous in the cyber defense context, it still imposes several challenges which I summarize into three aspects as the following:

First, even though there are off-the-shelf techniques, such as "incremental calculation of weighted mean and variance" [22], in the literature which satisfy memory and computation requirements, we need more sophisticated models to decay long-term dependencies and allow covariance matrices to be more related to recent time windows since we detect anomalies based on changes in network topographies as we have shown in Section 15.

Second, we can only perform Robust PCA on instances with numerically meaningful distance due to its optimization and matrix decomposition, and we have described how to support such a requirement by One-Hot encoding in Section 15. In

a streaming environment, One-Hot encoding should also be revised to the streaming fashion which is able to support dynamical modification on the encoding of novel measurements and changes on the dimension of covariance matrices.

Last but not least, it is also our goal to distinguish whether a new instance is a normal instance or an anomaly.

Herein, we propose a novel solution that tackles streaming anomaly detection problem by measuring the local correlation of data features in a *streaming fashion* instead of treating time windows as being independent. We also provide a reliable encoding algorithm that transforms categorical data to numerical as well as automatically enlarge covariance matrices when novel data arriving. Last, we describe an efficient strategy that projects new instances onto the space discovered by Robust PCA. It allows us to calculate anomaly scores for each new instance and reports abnormal measurements in real time.

17.1.1 Streaming Covariance Computation with Forget Rates

Due to the volume of cyber network data, our goal is to build memory-friendly algorithm. By developing a *streaming anomaly detection algorithm* based on computing sensor correlations in a streaming fashion and relaxing memory usage across time windows. In particular, our algorithm start with "incremental calculation of weighted mean and variance" [22] which only keeps mean value $\overline{x}_{n-1} \in \mathbb{R}^k$ and covariance $C_{n-1} \in \mathbb{R}^{k \times k}$ in the main memory, where k denotes the dimension of encoded data and $n-1$ represents the updated state after of the $(n-1)$th instance. The updating process consists of two steps. First we update the mean value with the new instance x_n:

$$\overline{x}_n = \overline{x}_{n-1} + \frac{x_n - \overline{x}_{n-1}}{n}$$

Next, we update the covariance C with result $\overline{x}_n$ of the first step:

$$C_n = C_{n-1} + (x_n - \overline{x}_n)(x_n - \overline{x}_{n-1}).$$

However, such an algorithm will eventually converge to the global correlation of features instead of local correlation. Our next step is to enhance the method with a forgetting mechanism. We are inspired by some key ideas of Kalman filter [65] on our covariance computation which accepts weighted temporal relation between measurements. In particular, the diagonal elements $e_{i,i}$ in covariance matrices indicate variance of ith features which usually are standardized to 1. Off-diagonal elements $e_{i,j}$ represent *linear* correlation between features [23]. An identity covariance matrix I represents linearly independent relationship between features. A summation of a non-identity correlation matrix and an identity correlation matrix emphasizes on independence between features and, simultaneously, reduces the correlations between them. Formally,

$$C' = C + \lambda I,$$

where C denotes a correlation matrix, I denotes an identity matrix, and C' denotes a correlation matrix that the off-diagonal elements depreciated. Therefore, the mutual influences between features in C' are weakened given a correlation matrix C and the feature correlation represented in C are forgotten in C'. We merge the "incremental calculation of weighted mean and variance" algorithm with this forgetting mechanism as follows:

Input: a list of instances x_1, x_2, ..., x_n, ... $x_n \in \mathbb{R}^k$ and λ.

Initialize $C \in R^{k \times k}$, $\overline{x}_{n-1} \in \mathbb{R}^k$, $\overline{x}_n \in \mathbb{R}^k$ to be zero matrices, and $i = 1$.

While(True):

1. Update mean value:

$$\overline{x_n} = \overline{x}_{n-1} + \frac{x_i - \overline{x}_{n-1}}{i}$$

2. Update covariance matrix C:

$$C = C + (x_i - \overline{x}_n) \cdot (x_i - \overline{x}_{n-1})$$

3. Storage covariance matrix C:

yield C

4. Update covariance matrix C with forgetting:

$$C = C + \lambda I$$

5. Move to next instance:

$$i = i + 1$$

$$\overline{x}_{n-1} = \overline{x}_n$$

17.1.2 Streaming Encoding

As described in Section 15, we apply One-Hot encoding to transform the categorical network data to numerical since Robust PCA can only work on numerical data [48]. In statistics, One-Hot encoding is often used for representing categorical data to dummy variables [23]. One-Hot encoding requires knowing the number of categories in advance, but it is not the case in streaming computation that new instances may bring novel categories which will cause failure of One-Hot encoding algorithm. In Figure 28, the original feature is shown on the left, where the instance marked "44330" is a new category. The One-Hot encoder needs to enlarge

the dimension of encoded data from 3 columns to 4.

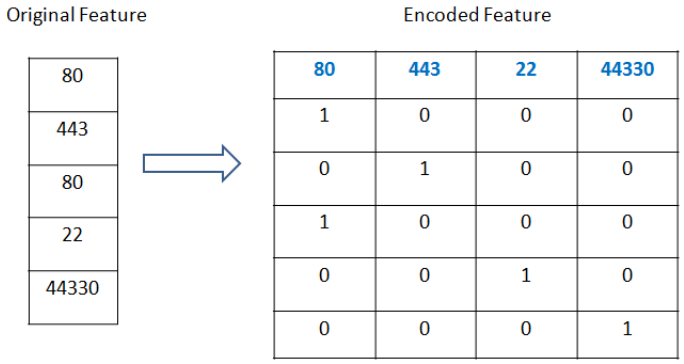

Figure 28: This figure shows the streaming One-Hot encoding must support dynamically changes on encoded features. The row of original feature "44330" is a novel category for encoder, and the encoder must support enlarge dimension of encoded data from 3 to 4.

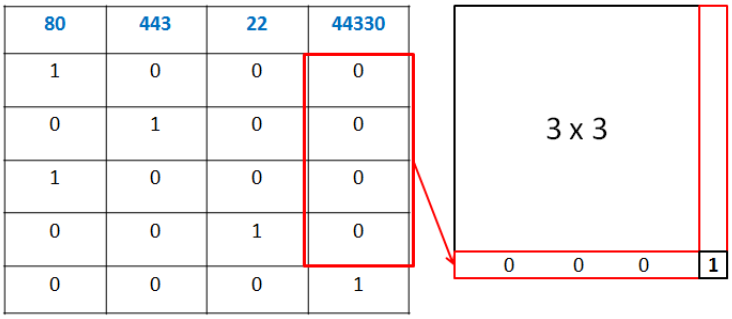

Figure 29: This figure shows how we extend the dimension of a covariance matrix when one new feature is novel. The new feature is linearly independent with existing features so off-diagonal elements are 0s and its variance is set to 1.

This method allows One-Hot encoding without the number of categories in advance which exactly matches the streaming process requirements. It also leaves the possibility to standardize data which will release covariance from influence of units and scales of variables, but it requires calculating the mean and variance of variables on the real-time.

17.1.3 Outlier Detection

Network traffic data is a typical example that anomalies are commonplace in the real-world. When we apply RPCA on normal traffic data, RPCA allows for the careful teasing apart of sparse anomalies so that the remaining low-dimensional approximation is faithful to the noise-free low-dimensional subspace describing normal traffic [48]. However, in PCA, this would cause the low-dimensional subspace to be as large as the anomalies. Unlike PCA, RPCA expose the faithful lower-dimensional subspace by removing outliers to the S matrix.

The property of discovering faithful low-dimensional subspace allow us to examining anomalies in the new data. The basic idea is when new instances X come, we project them onto the known space L which is discovered by our second order streaming analysis of RPCA. We compute a anomaly score by computing the difference Δ between any instance X and its projection X_p. We visualize this process in Figure 30. If a new sample if far away from the known L, then we can conclude that the new sample X is an outlier.

Further, we can also distinguish which features cause Δ being large by measuring the distance of a single feature to its projection. In particular, our method is detailed as follows:

Input: $Cor = X^T X \in \mathbb{R}^{k \times k}$ and $X_{new} \in \mathbb{R}^{1 \times k}$

1. Apply RPCA on the covariance matrix.

$V, \Sigma, V^T, S = RPCA(Cor)$

2. Compute the dimension of Cor by counting the number of non-zero diagonal elements of Σ*.*

dim = Count_non_zero(Σ)

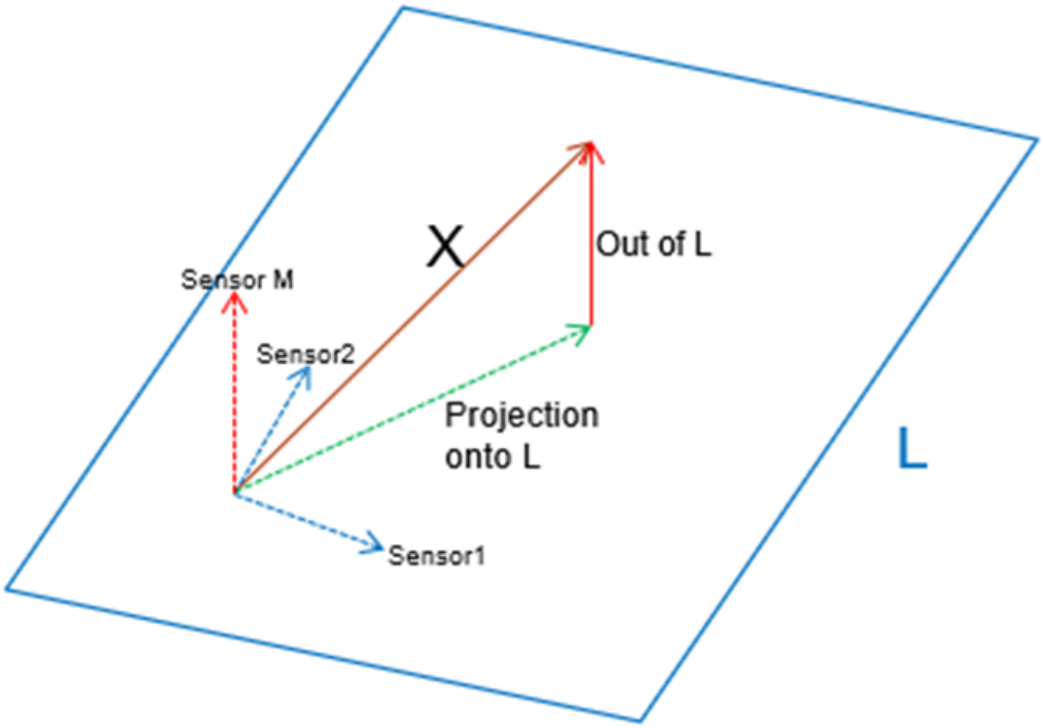

Figure 30: This figure shows our idea of projection. L represent the known space which we get from applying RPCA on known normal traffic. X is a new sample. We project X onto L and compute the difference between X and its projection.

3. *Compute the projection matrix V_p.*

   $V_p = V[0 : dim]$

4. *Project the new instance X_{new} onto the L.*

   $X_{proj} = X_{new} \cdot V_p \cdot V_p^T$

5. *Compute the difference.*

   $\Delta = X_{new} - X_{proj}$

17.2 Experimental Evaluation

To test our projection method, we describe the dataset and parameter settings for our experiments. In particular, we conduct experiments on both synthetic and real-world datasets. The synthetic data are generated by selecting a low-dimensional space which support sample normal data on. We set up a 20 low-dimensional subspace where we randomly generate 8000 instances with 100 features, but their true dimension is 20. We use the first 7999 samples as our training data while leave one out as the test. The training data is the input to RPCA to discover low-dimensional space.

In Figure 31, the main boxplot is computed by the difference between X and its projection, where we use the summation of absolute values of difference of each sample. The y-axis shows that overall scale of difference is at $1e-11$ the projection difference of normal points is fairly small. In Figure 31, we use the same normal data and its projection in the previous experiment but one random generated test sample. The red point shows the test sample have significant difference with normal which is the evidence that we mark it as an outlier. In Figure 33, we examine which features cause such big difference.

17.3 Section Summary

In this work, we develop a novel way to compute streaming covariance matrices that allows to decay long-term dependencies and result in covariance matrices being more related to recent time windows. Also, we revise One-Hot encoding to the streaming fashion which is able to support dynamical modification on the encoding of novel measurements and changes on the dimension of covariance matrices. Beyond the contributions above, we use the projection method to detect anomalies

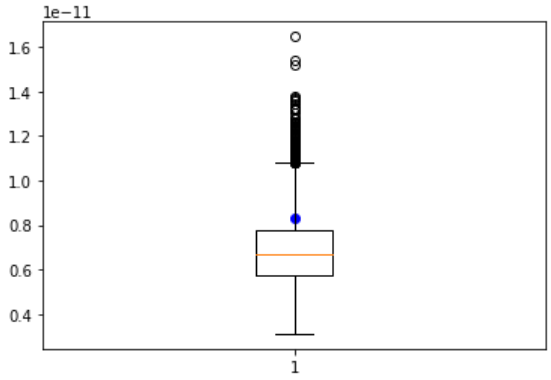

Figure 31: This figure shows the difference between X and its own projection. Each point is the summation of absolute values of difference of each sample. One can see that overall scale is at $1e-11$ the projection difference of normal points is fairly small.

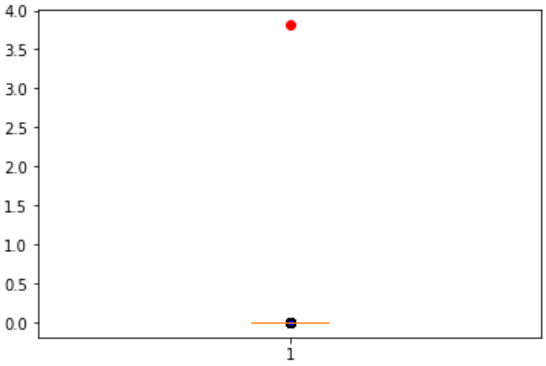

Figure 32: This figure shows the difference between X and an outlier. As the summation of absolute values of difference of each sample, the boxplot shows normal samples in X have small difference with its projection. However, the red point shows outlier will have significant difference with normal.

in new coming data.

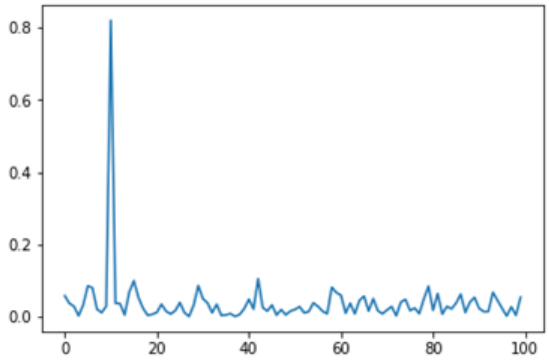

Figure 33: This figure show the difference of the outlier detected in Figure 32 at the feature level. The x-axis is the feature index while the y-axis is the difference level. It is clear to see that one feature (the 10th feature) has apparently higher difference than other which we believe result in this sample being marked as an outlier.

Part IV

Conclusion and Future Work

18 Conclusion

The work to date in this Ph.D. began by building on the foundations of Robust PCA and from this starting place we derived a family of robust methods that project the nominal data onto a low-dimensional manifold and filter out the exceptions as anomalies. We review state-of-the-art deep models and focus on deep autoencoders and sparse autoencoders which construct low-dimensional representations based on non-linear combinations of input features. We propose $\ell_{2,1}$ regularized sparse autoencoders which are more faithful to the idea of dimension reduction and such model is further used as a component in a robust sparse autoencoder model.

Inspired by denoising autoencoders, we create a new *"Robust Deep Autoen-*

coders" model that the case where no clean, noise-free data is available, These methods use an anomaly regularizing penalty based upon either ℓ_1 or $\ell_{2,1}$ norms. We also demonstrate how to use $\ell_{2,1}$ sparse autoencoders in the robust model framework and give rise to a new "Robust Sparse Autoencoders". We show how stacked denoising autoencoders can be generalized to detecting anomalies when there is no clean, noise-free data available, creating a new family of methods that we call *"Hierarchical Robust Deep Autoencoders (HRDA)"*.

In addition, we developed a training algorithm for the optimization problems of RDA and RSA of which we observe experimental convergence.

In the application aspect, we apply Robust PCA models on cyber security analysis where data is generated from Raytheon BBN's high fidelity data. We add an semi-supervised learning in which both Robust Principal Component Analysis (RPCA) and Robust Deep Autoencoder (RDA) on defenses of DDoS to provide unique features that reflect both overwhelming and rare traits of different DDoS attacks. From our experimental results, we conclude that both low-dimensional and anomaly features can identify certain kind of DDoS attacks.

Finally, we adapts out static anomaly detecting technique, Robust PCA, to streaming environments which is the able to identify abnormal instances while additional data is generated over time.

19 Future Directions

In addition, there are also some directions that are worthy to mention:

19.1 Adversarial Learning

Research is constantly pushing machine learning models to be faster, more accurate, and more efficient. However, an often overlooked aspect of designing and training models is security and robustness, especially in the face of an adversary who wishes to fool the model [33]. Recent literature is aware the security vulnerabilities of machine learning models. In fact, some researchers find that adding imperceptible perturbations to an image can cause drastically different model performance [33]. It is interesting that how well the RPCA framework, $X = L + S$, can individually resist such an adversarial attack and help existing vulnerable machine models.

19.2 Hybrid Attacks

Our applications on detecting DDoS attacks consider SYN-flood and Slowread attacks individually. Though typical, there exist close cousins to these two attacks, like SlowLoris, SlowReq, Slowdroid, and SlowComm to name but a few [2, 12, 13, 33]. A more detailed evaluation on these attacks have entered our consideration, especially we are look forward the performances on defending hybrid cyber attacks.

References

[1] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., ET AL. Tensorflow: a system for large-scale machine learning. In *OSDI* (2016), vol. 16, pp. 265–283.

[2] AIELLO, M., MONGELLI, M., CAMBIASO, E., AND PAPALEO, G. Profiling dns tunneling attacks with pca and mutual information. *Logic Journal of the IGPL 24*, 6 (2016), 957–970.

[3] BAUSCHKE, H. H., AND BORWEIN, J. M. On projection algorithms for solving convex feasibility problems. *SIAM review 38*, 3 (1996), 367–426.

[4] BERTHELOT, D., SCHUMM, T., AND METZ, L. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717* (2017).

[5] BERTSEKAS, D. P. *Convex optimization theory*. Athena Scientific Belmont, 2009.

[6] BISHOP, C. M. *Pattern recognition and machine learning*. springer, 2006.

[7] BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning 3*, 1 (2011), 1–122.

[8] BOYD, S., AND VANDENBERGHE, L. *Convex optimization*. Cambridge university press, 2004.

[9] BOYLE, J. P., AND DYKSTRA, R. L. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*. Springer, 1986, pp. 28–47.

[10] BUADES, A., COLL, B., AND MOREL, J.-M. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation 4*, 2 (2005), 490–530.

[11] BUCZAK, A. L., AND GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials 18*, 2 (2016), 1153–1176.

[12] CAMBIASO, E., PAPALEO, G., AND AIELLO, M. Slowdroid: Turning a smartphone into a mobile attack vector. In *2014 International Conference on Future Internet of Things and Cloud* (2014), IEEE, pp. 405–410.

[13] CAMBIASO, E., PAPALEO, G., AND AIELLO, M. Slowcomm: Design, development and performance evaluation of a new slow dos attack. *Journal of Information Security and Applications 35* (2017), 23–31.

[14] CAMBIASO, E., PAPALEO, G., CHIOLA, G., AND AIELLO, M. Slow dos attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications 1*, 3-4 (2013), 300–319.

[15] CANDÈS, E. J., LI, X., MA, Y., AND WRIGHT, J. Robust principal component analysis? *Journal of the ACM (JACM) 58*, 3 (2011), 11.

[16] CERF, V., AND KAHN, R. A protocol for packet network intercommunication. *IEEE Transactions on communications 22*, 5 (1974), 637–648.

[17] CHEN, J., CHEN, J., CHAO, H., AND YANG, M. Image blind denoising with generative adversarial network based noise modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3155–3164.

[18] COMBETTES, P. L., AND PESQUET, J.-C. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[19] DENTON, E. L., CHINTALA, S., FERGUS, R., ET AL. Deep generative image models using a? laplacian pyramid of adversarial networks. In *Advances in neural information processing systems* (2015), pp. 1486–1494.

[20] DOERSCH, C. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

[21] DONOHO, D. L. For most large underdetermined systems of linear equations the minimal 1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics 59*, 6 (2006), 797–829.

[22] FINCH, T. Incremental calculation of weighted mean and variance. *University of Cambridge 4* (2009), 11–5.

[23] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, NY, USA:, 2001.

[24] GEHRING, J., MIAO, Y., METZE, F., AND WAIBEL, A. Extracting deep bottleneck features using stacked auto-encoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 3377–3381.

[25] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep learning*. MIT press, 2016.

[26] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[27] HEUSEL, M., RAMSAUER, H., UNTERTHINER, T., NESSLER, B., AND HOCHREITER, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems* (2017), pp. 6626–6637.

[28] HU, W., LIAO, Y., AND VEMURI, V. R. Robust support vector machines for anomaly detection in computer security. In *ICMLA* (2003), pp. 168–174.

[29] JAMES, G., WITTEN, D., HASTIE, T., AND TIBSHIRANI, R. *An Introduction to Statistical Learning*, vol. 6. Springer, 2013.

[30] KAUR, S., AND SINGH, N. Image denoising techniques: A review. *International Journal of Innovative Research in Computer and Communication Engineering 2*, 6 (2014).

[31] KINGMA, D. P., AND WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[32] KULLBACK, S., AND LEIBLER, R. A. On information and sufficiency. *The annals of mathematical statistics 22*, 1 (1951), 79–86.

[33] KURAKIN, A., GOODFELLOW, I., BENGIO, S., DONG, Y., LIAO, F., LIANG, M., PANG, T., ZHU, J., HU, X., XIE, C., ET AL. Adversarial attacks and defences competition. 195–231.

[34] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[35] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[36] LECUN, Y., CORTES, C., AND BURGES, C. J. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1998.

[37] LEDIG, C., THEIS, L., HUSZÁR, F., CABALLERO, J., CUNNINGHAM, A., ACOSTA, A., AITKEN, A., TEJANI, A., TOTZ, J., WANG, Z., ET AL. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), IEEE, pp. 105–114.

[38] LEE, H., BATTLE, A., RAINA, R., AND NG, A. Y. Efficient sparse coding algorithms. In *Advances in neural information processing systems* (2007), pp. 801–808.

[39] LIU, F. T., TING, K. M., AND ZHOU, Z.-H. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on* (2008), IEEE, pp. 413–422.

[40] LUCIC, M., KURACH, K., MICHALSKI, M., GELLY, S., AND BOUSQUET, O. Are gans created equal? a large-scale study. In *Advances in neural information processing systems* (2018), pp. 698–707.

[41] LYUDCHIK, O. Outlier detection using autoencoders. Tech. rep., 2016.

[42] MA, Y., ZHANG, P., CAO, Y., AND GUO, L. Parallel auto-encoder for efficient outlier detection. In *Big Data, 2013 IEEE International Conference on* (2013), IEEE, pp. 15–17.

[43] MAMMADOV, S., MEHTA, D., STONER, E., AND CARVALHO, M. M. High fidelity adaptive cyber emulation. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (2017), 1–8.

[44] MENG, L., DING, S., AND XUE, Y. Research on denoising sparse autoencoder. *International Journal of Machine Learning and Cybernetics 8*, 5 (2017), 1719–1729.

[45] MOSCI, S., ROSASCO, L., SANTORO, M., VERRI, A., AND VILLA, S. Solving structured sparsity regularization with proximal methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2010), Springer, pp. 418–433.

[46] NG, A. Sparse autoencoder. *CS294A Lecture notes 72*, 2011 (2011), 1–19.

[47] NIELSEN, M. A. *Neural networks and deep learning*, vol. 25. Determination press USA, 2015.

[48] PAFFENROTH, R., DU TOIT, P., NONG, R., SCHARF, L., JAYASUMANA, A. P., AND BANDARA, V. Space-time signal processing for distributed pattern detection in sensor networks. *IEEE Journal of Selected Topics in Signal Processing 7*, 1 (2013), 38–49.

[49] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., ET AL. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12*, Oct (2011), 2825–2830.

[50] QI, Y., WANG, Y., ZHENG, X., AND WU, Z. Robust feature learning by stacked autoencoder with maximum correntropy criterion. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (2014), IEEE, pp. 6716–6720.

[51] RADFORD, A., METZ, L., AND CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[52] REED, S., AKATA, Z., YAN, X., LOGESWARAN, L., SCHIELE, B., AND LEE, H. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396* (2016).

[53] REN, H., YUE, Y., ZHOU, C., PAFFENROTH, R. C., AND LI, Y. Robust variational autoencoders.

[54] REZENDE, D. J., MOHAMED, S., AND WIERSTRA, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

[55] RIFAI, S., VINCENT, P., MULLER, X., GLOROT, X., AND BENGIO, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), pp. 833–840.

[56] RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J., ET AL. Learning representations by back-propagating errors. *Cognitive modeling 5*, 3 (1988), 1.

[57] SANGWINE, S. J., AND HORNE, R. E. *The colour image processing handbook*. Springer Science & Business Media, 2012.

[58] TRIPATHI, S., LIPTON, Z. C., AND NGUYEN, T. Q. Correction by projection: Denoising images with generative adversarial networks. *arXiv preprint arXiv:1803.04477* (2018).

[59] VINCENT, P., LAROCHELLE, H., BENGIO, Y., AND MANZAGOL, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 1096–1103.

[60] VINCENT, P., LAROCHELLE, H., LAJOIE, I., BENGIO, Y., AND MANZAGOL, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research 11*, Dec (2010), 3371–3408.

[61] XIAO, H., RASUL, K., AND VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[62] XIE, J., XU, L., AND CHEN, E. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems* (2012), pp. 341–349.

[63] YAN, X., YANG, J., SOHN, K., AND LEE, H. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision* (2016), Springer, pp. 776–791.

[64] YANG, Q., YAN, P., ZHANG, Y., YU, H., SHI, Y., MOU, X., KALRA, M. K., ZHANG, Y., SUN, L., AND WANG, G. Low dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE transactions on medical imaging* (2018).

[65] ZARCHAN, P., AND MUSOFF, H. *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, Inc., 2013.

[66] ZARGAR, S. T., JOSHI, J., AND TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials 15*, 4 (2013), 2046–2069.

[67] ZHANG, J., ZULKERNINE, M., AND HAQUE, A. Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38*, 5 (2008), 649–659.

[68] ZHAO, D., GUO, B., WU, J., NING, W., AND YAN, Y. Robust feature learning by improved auto-encoder from non-gaussian noised images. In *Imaging Systems and Techniques (IST), 2015 IEEE International Conference on* (2015), IEEE, pp. 1–5.

[69] ZHOU, C., AND PAFFENROTH, R. C. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), ACM, pp. 665–674.

[70] ZHU, J.-Y., KRÄHENBÜHL, P., SHECHTMAN, E., AND EFROS, A. A. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision* (2016), Springer, pp. 597–613.