Anomaly Detection with Robust Deep Autoencoders

Chong Zhou
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
czhou2@wpi.edu

Randy C. Paffenroth
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
rcpaffenroth@wpi.edu

ABSTRACT

Deep autoencoders, and other deep neural networks, have demonstrated their effectiveness in discovering non-linear features across many problem domains. However, in many real-world problems, large outliers and pervasive noise are commonplace, and one *may not have access to clean training data* as required by standard deep denoising autoencoders. Herein, we demonstrate 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*. Our model is inspired by Robust Principal Component Analysis, and we split the input data X into two parts, $X = L_D + S$, where L_D can be effectively reconstructed by a deep autoencoder and S contains the outliers and noise in the original data X. Since such splitting increases the robustness of standard deep autoencoders, we name our model a "Robust Deep Autoencoder (RDA)". Further, we present generalizations of our results to grouped sparsity norms which allow one to distinguish random anomalies from other types of structured corruptions, such as a collection of features being corrupted across many instances or a collection of instances having more corruptions than their fellows. Such "Group Robust Deep Autoencoders (GRDA)" give rise to novel anomaly detection approaches whose superior performance we demonstrate on a selection of benchmark problems.

KEYWORDS

Autoencoders; Robust Deep Autoencoders; Group Robust Deep Autoencoder; Denoising; Anomaly Detection

1 INTRODUCTION

Deep learning is part of a broad family of methods for representation learning [11], and it has been quite successful in pushing forward the state-of-the-art in multiple areas [9–11, 29]. In particular, herein we focus on deep autoencoders which construct representations based on non-linear combinations of input features, and where the non-linearity is classically introduced by way of some non-linear activation function [11]. Unfortunately, outliers and noise may reduce the quality of representations discovered by deep autoencoders [15, 16]. There is a large extent literature which attempts to address this challenge and two approaches include *denoising autoencoders* and *maximum correntropy autoencoders*. Denoising autoencoders [9, 17, 23, 30, 31], require access to a source of clean, noise-free data for training, and such data is not always readily available in real-world problems [28]. On the other hand, maximum correntropy autoencoders replace the reconstruction cost with a noise-resistant criteria *correntropy* [22]. 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.

As we will detail, 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.

1.1 Contribution

Herein we show how denoising autoencoders can be generalized to the case where no clean, noise-free data is available, and we demonstrate the superior performance of our proposed methods using standard benchmark problems such as the MNIST data set [12]. We call such algorithms *"Robust Deep Autoencoders (RDA)"*, and our proposed models improve upon normal deep autoencoders by introducing an anomaly regularizing penalty based upon either ℓ_1 or $\ell_{2,1}$ norms. Using such a regularizing penalty, we derive a training algorithm for the proposed model by combining ideas from proximal methods [4], backpropagation [24], and the Alternating Direction of Method of Multipliers (ADMM) [3]. As an interesting consequence of the development of such RDAs, we have also derived a novel family of unsupervised anomaly detection algorithms. We demonstrate the effectiveness of these anomaly detection algorithm, as compared to a baseline approach, on a number of challenging benchmark problems.

In particular, we emphasize that our proposed methods differ from standard techniques in two important ways.

- First, our methods *differ from standard denoising autoencoders* [28] in that they do not require any noise-free training data. Rather, our methods compute the entries on which the autoencoder loss function is enforced *on-the-fly* based upon the performance of the underlying autoencoder algorithm.
- Second, our methods *differ from standard sparse autoencoders* [13] in that we do not penalize the sparsity of the hidden layer. Rather, our methods allow for a sparse set

KDD '17, August 13-17, 2017, Halifax, NS, Canada

978-1-4503-4887-4/17/08…$15.00
DOI: 10.1145/3097983.3098052

of *exceptions* to the enforcement of the autoencoder loss function.

2 BACKGROUND

We begin the derivation of our methods by providing a brief introduction to non-linear feature representation by way of deep autoencoders. We also present a few foundational ideas from Robust Principal Component Analysis (RPCA) which are classically used to construct low-dimensional *linear* features by filtering out outlying measurements using a convex relaxation of the rank operator [6, 20].

2.1 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 have no reconstruction error. 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 a *low-dimensional*, non-linear representation of the input data.

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

$$\overline{X} = D(E(X)) \tag{1}$$

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:

$$\min_{D,E} \|X - D(E(X))\| \tag{2}$$

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

Usually, an autoencoder with more than one hidden layers is called a deep autoencoder [11]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 represent complicated distributions over the input X. In the sequel, our focus will be on deep autoencoders with all autoencoders assumed to be deep.

2.2 Robust Principal Component Analysis

Robust Principal Component Analysis (RPCA) [6, 7, 20] is a generalization of Principal Component Analysis (PCA) [8] that attempts to reduce the sensitivity of PCA to outliers. In particular, RPCA allows for the careful teasing apart of sparse outliers so that the remaining low-dimensional approximation is faithful to the noise-free low-dimensional subspace describing the bulk of the raw data [6, 7, 20].

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

$$X = L + S. \tag{3}$$

The matrix L contains a low-dimensional representation of X [6, 7, 20] and the matrix S consists of element-wise outliers, which can not be efficiently captured by the low-dimensional representation L.

Naively, this matrix decomposition can be computed by way of the following optimization problem [6, 7]

$$\begin{aligned} &\min_{L,S} \rho(L) + \lambda||S||_0 \\ &\text{s.t. } \|X - L - S\|_F^2 = 0, \end{aligned} \tag{4}$$

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 (4) is NP-hard [6, 7] and only computationally tractable for small matrices X. However, there is a large literature [6, 7, 20] on convex relaxations of this class of problems, such as

$$\begin{aligned} &\min_{L,S} ||L||_* + \lambda||S||_1 \\ &\text{s.t. } \|X - L - S\|_F^2 = 0, \end{aligned} \tag{5}$$

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).

While optimization (4) is NP-hard, optimization (5) is convex and can be solved for large data matrices X. With this idea, the key insight of our method can be succinctly described. It is well known [10] that, when one considers a Frobenius loss function and constrains the autoencoder D and E (2) to be linear maps, that PCA is an optimal autoencoder. *Accordingly, our method can be viewed as replacing the nuclear norm in (5), which results in a linear projection to a low-dimensional hidden layer, with a non-linear autoencoder, which results in a non-linear projection to a low-dimensional hidden layer.*

Of course, one must therefore solve (5) but with a more complicated objective function. However, using ideas such as the Alternating Direction Method of Multipliers [3], we demonstrate how one can address such problems by combining standard off-the-shelf solvers for optimizing a deep autoencoder with standard techniques for optimizing proximal problems involving the ℓ_1 or other similar penalties [4, 20].

3 METHODOLOGY

In this section we provide the derivation of our proposed methodologies for constructing *Robust Deep Autoencoders*, which we view as a combination of autoencoders and robust principal component analysis. The central idea is that a 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[1] 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 into S, the remaining data L_D

[1] In this paper, the term "noise" implies the element-wise noise, the "outlier" implies the row-wise outlying instance and the "anomalies" implies both.

can be accureately reconstructed by an autoencoder with such a low-dimensional hidden layer.

3.1 Robust Deep Autoencoders with ℓ_1 Regularization

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.

Similar to RPCA, a Robust Deep Autoencoder 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

$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_0 \quad \text{s.t. } X - L_D - S = 0, \tag{6}$$

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.

Foreshadowing our results in Section 5, we note that 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 (6) is not computationally tractable, just as is (4). However, following the RPCA literature [6, 7, 20], one can relax the combinatorial $\|S\|_0$ term of the optimization (6) by replacing it with a convex relaxation $\|S\|_1$, giving rise to the optimization

$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_1 \quad \text{s.t. } X - L_D - S = 0. \tag{7}$$

Notice that in the constraint of (7) 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 $D_\theta(E_\theta(\cdot))$.

Note, while 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

$$E_\theta(x) = E_{W,b}(x) = \text{logit}(Wx + b_E)$$

and

$$D_\theta(x) = D_{W,b}(x) = \text{logit}(W^T E_{W,b}(x) + b_D)$$

[11].

3.2 Robust Deep Autoencoders with $\ell_{2,1}$ Regularization

Our basic RDA in Section 3.1 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 1.

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 1. 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 [2].

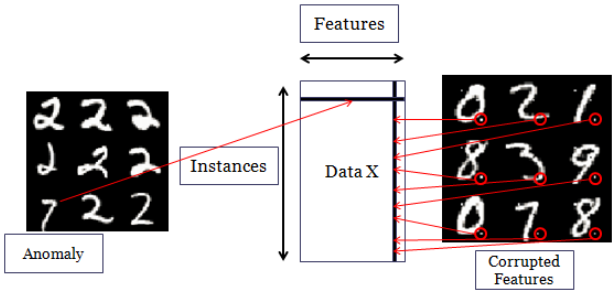

Figure 1: This figure shows two examples of group anomalies. For example, on the left, we see a collection of 2s corrupted by a single 7. This corresponds to a row of X being anomalous. Similarly, on the right, we see a collection of digits where a particular pixel is always on. This corresponds to a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

3.3 Anomalous Feature and Instance Detection

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

$$||X||_{2,1} = \sum_{j=1}^{n} ||x_j||_2 = \sum_{j=1}^{n} (\sum_{i=1}^{m} |x_{ij}|^2)^{1/2} \tag{8}$$

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 [19]. 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 (8) in mind, we can pose the following optimization problem

$$\min_{\theta,S} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_{2,1} \quad \text{s.t. } X - L_D - S = 0, \tag{9}$$

where the terms are defined as in (7) except the ℓ_1 norm has been replaced by the grouped norm $\ell_{2,1}$.

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:

$$\min_{\theta,S} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S^T\|_{2,1} \quad \text{s.t. } X - L_D - S = 0. \tag{10}$$

4 ALGORITHM TRAINING

In this section we provide the details of our methodology for solving problems such as (7), (9), and (10). In particular, we are inspired by ideas such as the Alternating Direction Method of Multipiers [4] and R. L. Dykstra's alternating projection method [5][2]. As we will detail, we iteratively optimize each of the terms in (7), (9), and (10) separately, interspersed with projections onto the constraint manifold. Note, 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 [29]. However, as we will demonstrate in Section 5, we have substantial empirical evidence that the optimization algorithm we present below provides high-quality solutions.

4.1 Alternating Optimization for ℓ_1 and $\ell_{2,1}$ Robust Deep Autoencoder

The key insight of the Alternating Direction Method of Multipliers (ADMM) [4] 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 [6, 7, 20] we split the objective in (7), (9), and (10) into a first piece which depends on L_D and is independent of S, namely $||L_D - D_\theta(E_\theta(L_D))||_2$, and a second piece which depends on S and is independent of L_D.

Back-propagation [10] is a typical training algorithm for deep learning and it is precisely the method we use for optimizing the term $||L_D - D_\theta(E_\theta(L_D))||_2$ when S is held constant. However, the terms such as $||S||_1$, $||S||_{2,1}$, and $||S^T||_{2,1}$ are more delicate since they are not differentiable. However, such non-differentiable terms can be phrased as proximal gradient problems [4], whose solution we will detail in Sections 4.2 and 4.3. To combine these two parts, we follow the idea of an ADMM [3, 4] and Dykstra's alternating projection method [5]. In effect, we use a back-propagation method to train an autoencoder to minimize $||L_D - D(E(L_D))||_2$ with S fixed, a proximal gradient to minimize 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: $X \in R^{N\times n}$
Initialize $L_D \in R^{N\times n}, S \in R^{N\times n}$ to be zero matrices, $LS = X$, and an 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 first term $||L_D - D(E(L_D))||_2$ using back-propagation.
3. Set L_D to be the reconstruction from the trained 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

4.2 Proximal Method for ℓ_1 Norm

The ℓ_1 norm can be optimized efficiently through the use of a proximal method [3, 4] such as

$$prox_{\lambda,l_1}(x_i) = \begin{cases} x_i - \lambda, & x_i > \lambda \\ x_i + \lambda, & x_i < -\lambda \\ 0, & x_i \in [-\lambda, \lambda]. \end{cases} \tag{11}$$

Such a function is known as a *shrinkage* operator and is quite common in ℓ_1 optimization problems. For additional details see [4].

The following pseudo-code provides an implementation of proximal operator $prox_{\lambda,l_1}(S)$:

Input: $S \in R^{m\times n}$, λ

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

```
For i in 1 to m × n:
    if S[i] > λ :
        S[i] = S[i] − λ
        continue
    if S[i] < −λ :
        S[i] = S[i] + λ
        continue
    if −λ <= S[i] <= λ :
        S[i] = 0
        continue
Return S
```

4.3 Proximal Method for $\ell_{2,1}$ Norm

Similar to (11) 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 [3, 18, 19]

$$(prox_{\lambda, l_{2,1}}(x))^j = \begin{cases} x_g^j - \lambda \dfrac{x_g^j}{||x_g||_2}, & ||x_g||_2 > \lambda \\ 0, & ||x_g||_2 <= \lambda, \end{cases} \tag{12}$$

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 (11) becomes block-wise sparsity. For additional details see [4].

The following pseudo-code provides an implementation of proximal operator $prox_{\lambda, l_{2,1}}(S)$:

```
Input: S ∈ R^{m×n}, λ
For j in 1 to n:
    e_j = ( Σ_{i=1}^{m} |S[i,j]|^2 )^{1/2}
    if e_j > λ :
        For i in 1 to m:
            S[i,j] = S[i,j] − λ S[i,j] / e_j
        continue
    if e_j <= λ :
        For i in 1 to m:
            S[i,j] = 0
        continue
Return S
```

5 NUMERICAL RESULTS

In this section we demonstrate the effectiveness of our propose robust autoencoders using the well-known image recognition MNIST data set [12]. The training set consists of 50,000 instances, and each instance is a 28×28 pixel image. Each image is labeled as a digit ranging from "0" to "9". However, as we will detail in the sequel, rather than merely considering the original MNIST data set, we corrupt the base images with noise and outliers to demonstrate and validate the various capabilities of our approaches.

5.1 Implementation Details

We build our implementation of a RDA using the Google machine learning library, Tensorflow, version 0.12.0rc0 [1] and Theano version 0.9 [26]. We ran our experiments on a virtual machine running an Ubuntu 14.04.3 image with 2 CPUs, and 4 GB memory. The image is run on hardware provided the NSF Jetstream Cloud [25, 27].

As is standard in the literature [11], we specify the encoder $E(\cdot)$ of each layer as $E(X) = g(X \cdot W + b_E)$, where X is our input data, g is an activation function, W is a projection from the input dimension to a lower dimensional hidden layer and b is a bias term. The decoder $D(\cdot)$ of each decoding layer is $\overline{X} = D(X) = g(E(X) \cdot W^T + b_D)$, where W^T projects a low-dimensional hidden layer back to the higher input dimension. We choose $g(\cdot)$ to be the sigmoid function $g(t) = \dfrac{1}{1 + \exp^{-t}}$.

The TensorFlow based autoencoder is the building block of our implementation and makes use of the following pseudo-code interface:

```
Initialize an autoencoder with sizes of each layer
    .init(layer_sizes)
Training the autoencoder, given data X
    .fit(X, iter, batch_size)
After training, get the hidden layer value
    .transform(X)
```

Our code is available on Github at: https://github.com/zc8340311/RobustAutoencoder/tree/master/lib

5.2 ℓ_1 Robust Deep Autoencoder for Denoising

In Figure 2, we show the key results for our proposed ℓ_1 penalized Robust Deep Autoencoder as compared against a standard autoencoder. Our experiment 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.

Then, these corrupted images are used to train a normal autoencoder and a RDA each with exactly the same number and breadth of layers. In particular, we use two hidden layers which project the input data from 784 dimensions to 196 dimensions, and then project 196 dimensions to 49 dimensions.

Both models are only trained on corrupted images and neither of them has any extra information, e.g. the labels of the images or any clean images. After training, we extract the dimension reduced features from their hidden 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. We split the data with 1/3 of the instances as a testing set and the remaining 2/3 as a training set. The quality of features produced by the various autoencoders are judged by the prediction accuracy of the classifier on the test set. For our supervised classifier, we choose a random forest as implemented in the scikit-learn package, version 0.18, [21]. We use 50 trees in the forest and leave the remaining

parameters for the classifier at their default values (e.g., each tree has access to a number features equal to the square root of the total number of input features). Note, we do not choose a state-of-the-art supervised learning method for this problem since the supervised learning method is merely used to assess the performance of the features provided by the (robust) autoencoder's hidden layers. We leverage the random forest to offer a fair comparison environment, which is consistent through all experiments.

For a fixed supervised classifier, we presume that higher test error rates are indicative of lower feature quality. With the above details in mind, Figure 2 demonstrates how the features provided by an autoencoder's hidden layer are corrupted by the noise in the images, while the robust autoencoder's hidden layers are more faithful representations of the original noiseless images, even though the RDA was trained using *only corrupted imagery*.

In the Figure 2, 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 2 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 2, 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 2 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 2, neither model can produce high-quality features and the testing accuracy of both methods is again the same.

5.3 $\ell_{2,1}$ Robust Deep Autoencoder for Outlier Detection

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%. Our experimental setup is shown in Figure 3.

We train an $\ell_{2,1}$ RDA using the mixed data without providing any side information (such as which images are actually "4"). It is our goal that the $\ell_{2,1}$ RDA should automatically pick out the high reconstruction error anomalies and isolate these outliers into S. The performances of the $\ell_{2,1}$ RDA can then be assessed by comparing

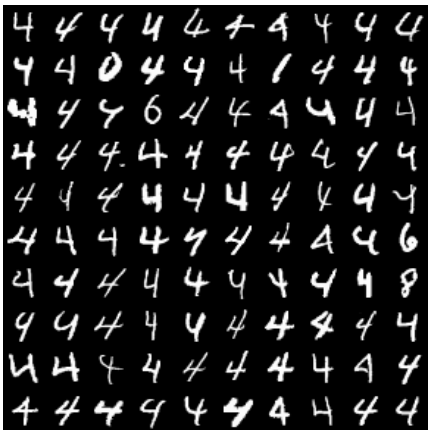

Figure 3: This figure shows the first 100 image instances for our outlier detection task. In this experiment, the "4" digits are nominal, and all other digits are considered as anomalies.

the predicted outlying instances in S and the true outliers whose label we know.

In this instance, the layer sizes of the RDA are 784, 400, and 200. [3] To get the optimal value of λ, we search a range from $1e-05$ to $1.5e-04$ with layer sizes fixed and only judge the λ value based on its F1-score for detecting outliers [8]. Figure 4 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 shows the results of our anomaly detection experiment. We wish to emphasize that the entire experiment is unsupervised, except for the training of *the single parameter* λ.

In particular, the training of the $\ell_{2,1}$ RDA is fully unsupervised, and the $\ell_{2,1}$ RDA is only trained with unlabeled images. Tuning the λ parameter makes the algorithm semi-supervised in that we use the F1-score to select the optimal λ value. The idea is that the optimal choice for λ does not revolve around the particular form of the anomalies. Rather, the choice of λ revolves around ensuring that all nominal instances are represented by L_D. In other words, λ is not chosen based on the anomalous instances, it is chosen based on the normal instances.

As shown in Figure 4 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. 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 [14]. The key idea of Isolation forests is that anomalies are

[3] Note, for this experiment we do not use a normal autoencoder.

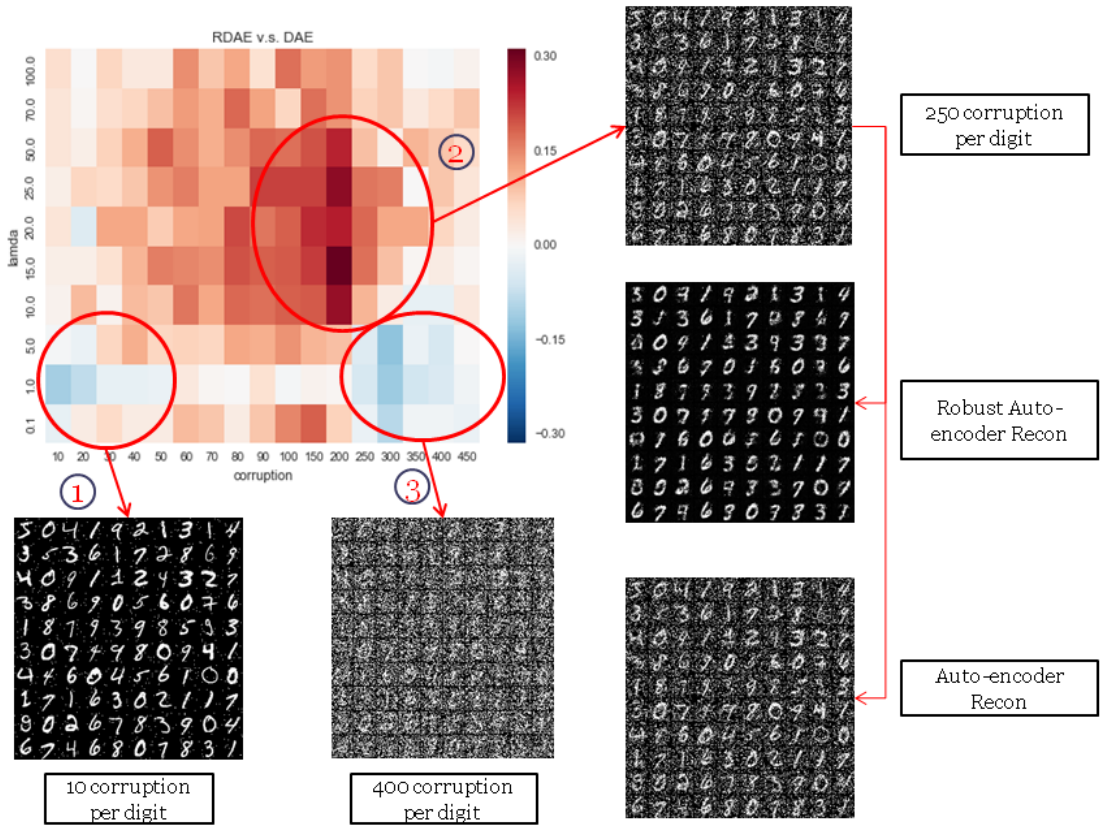

Figure 2: This figure shows the difference between error rates for the features constructed by a normal autoencoder and a RDA. Red indicates where the error rates of the RDA are superior to those of the normal autoencoder, and blue indicates where the converse is true. The errors rates are bases upon the prediction of image labels with respect to different degrees of corruption and different values of λ in the Robust Deep Autoencoders. The x-axis shows different corruption levels, and the y-axis shows different λ values. The two images on the bottom, indicated with ① and ③, show examples of images with the designated amount of corrupted pixels. 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 auto-encode performs up to 30% better. 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, while the images in the middle, produced from the Robust Deep Autoencoder'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.

'few and different', which make them more susceptible to isolation than normal points [14].

We use the Isolation Forests model as implemented by version 0.18.1 of the scikit-learn package [21]. The isolation forest model take two variables: the number 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 5 and 6, we can see the RDA gets as 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.

5.4 Convergence of Training Method

As we mentioned in Section 4, since the objective function of both an ℓ_1 RDA and an $\ell_{2,1}$ RDA is 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

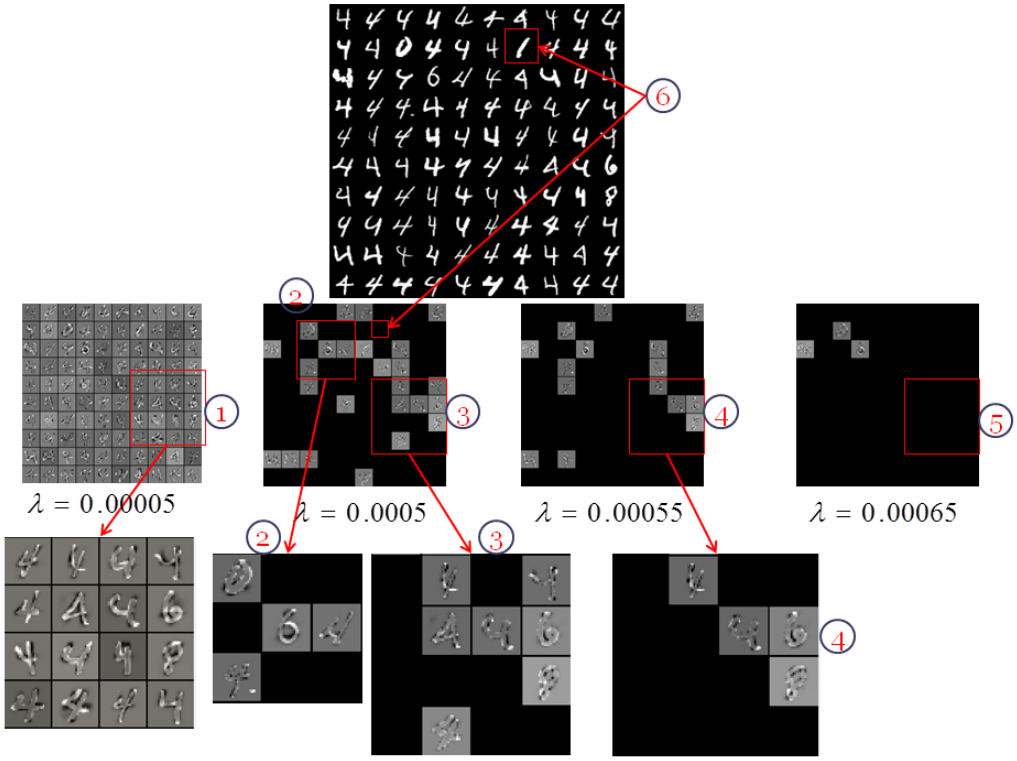

Figure 4: This figure shows how the sparsity of S changes with different λ values. ① shows a small λ, namely $\lambda = 0.00005$, which levies a small penalty on S, and the robust autoencoder emphasizes minimizing the reconstruction error by marking many images as anomalous. Accordingly, S is dense and 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.

our observations. 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 7, 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.

As we discussed in Section 3, λ is also essential to the convergence of the problem. Accordingly, Figure 7 shows convergence histories for several values of λ. As one can see, our proposed ADMM algorithm converges quickly in many cases of interest. A theoretical justification of these convergence properties will be a subject of future work.

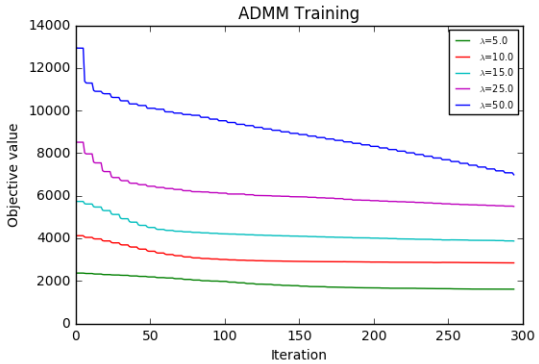

Figure 7: This figure shows the convergence of our optimization algorithm for (7). The objective descends quickly in the first 50 iterations and archives convergences after 200 iterations when λ is small. Convergence is slower for large λ values but still reasonable for our problems of interest.

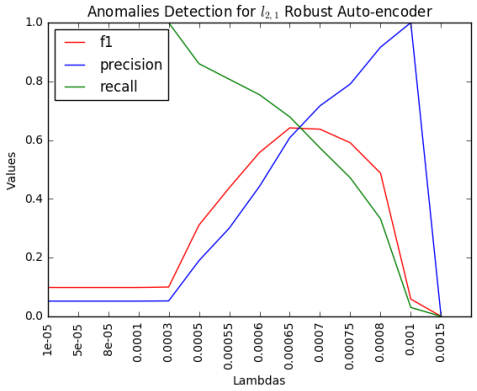

Figure 5: This figure shows the precision, recall and F1-scores with different λ values for the RDA. In the cases of a small λ, every row is non-zero and all the rows are marked as anomalies. We get high recall scores but very low precision and F1-scores. As the λ values increase, the F1-score and score increase reaching a maximum at $\lambda = 0.00065$ with an F1-score of 0.64.

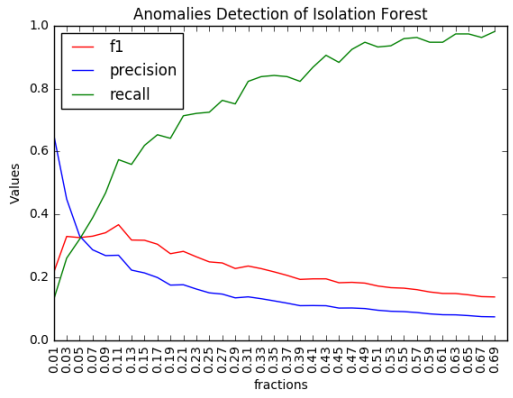

Figure 6: This figure shows the precision, recall and F1-score as we vary the fraction ratio for the Isolation Forest algorithm from 0.01 to 0.69. The optimal F1-score that the Isolation Forest achieves is about 0.37 when the fraction is equal to 0.11. This F1-score is approximately 73.0% worse than the score achieve by the RDA.

6 CONCLUSION

In this paper, 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 Autoencoders"*. These methods use an anomaly regularizing penalty based upon either ℓ_1 or $\ell_{2,1}$ norms. The resulting optimization problem was solved using ideas from proximal methods [4], backpropagation [24], and the Alternating Direction of Method of Multipliers (ADMM) [3]. We demonstrated the effectiveness of our approaches based upon noisy versions of the MNIST data set [12], where we achieved an approximately 30% improvement over standard autoencoders. As an interesting consequence of the development of such robust deep autoencoders, we also derived a novel family of unsupervised anomaly detection algorithms (parameterized by λ) and the effectiveness of these methods was demonstrated by comparing against Isolation Forests where we achieved a 73% improvement over that method. It is worth mentioning that in this paper we focus on improving autoencoders, however the $X = L_D + S$ framework, which is inspired by the RPCA, could be generalized where the autoencoder part is exchanged for other cost functions.

In the future, we would like to test our model on additional data sets from multiple areas, especially for the detection of cyber-attacks in network data. In addition, while we have substantial empirical evidence as to the convergence of our training algorithm for constructing high quality solutions, we would like to derive a theoretical basis for our training algorithm including convergence rates.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. (mar 2016). arXiv:1603.04467 http://arxiv.org/abs/1603.04467

[2] Christopher M Bishop. 2006. Pattern recognition. *Machine Learning* 128 (2006), 1–58.

[3] Stephen Boyd. 2010. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers slide (Alternating Direction Method of Multipliers). (2010). http://dl.acm.org/citation.cfm?id=2185816

[4] S.P. Boyd and L. Vandenberghe. 2004. *Convex optimization*. Vol. 25. DOI:http://dx.doi.org/10.1080/10556781003625177

[5] James P Boyle and Richard L Dykstra. 1986. A method for finding projections onto the intersection of convex sets in Hilbert spaces. In *Advances in order restricted statistical inference*. Springer, 28–47.

[6] E J Candès, X Li, Y Ma, and J Wright. 2009. Robust Principal Component Analysis? *Preprint: arXiv:0912.3599* (2009).

[7] David L. Donoho. 2006. 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. DOI: http://dx.doi.org/10.1002/cpa.20132 arXiv:0912.3599

[8] J. Friedman, T. Hastie, and R. Tibshirani. 2008. *The Elements of Statistical Learning*. Vol. 2. http://www-stat.stanford.edu/

[9] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. 2013. Extracting deep bottleneck features using stacked auto-encoders. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2013), 3377–3381. DOI:http://dx.doi.org/10.1109/ICASSP.2013.6638284

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. (2016). DOI:http://dx.doi.org/10.1038/nmeth.3707 arXiv:arXiv:1312.6184v5

[11] Y LeCun, Y Bengio, and G Hinton. 2015. Deep learning. *Nature* (2015). http://www.nature.com/nature/journal/v521/n7553/abs/nature14539.html

[12] Y LeCun, L Bottou, and Y Bengio. 1998. Gradient-based learning applied to document recognition. *Proceedings of the* (1998). http://ieeexplore.ieee.org/abstract/document/726791/

[13] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. 2007. Efficient sparse coding algorithms. *Advances in neural information processing systems* 19 (2007), 801.

[14] Fei Tony Liu and Kai Ming Ting. 2008. Isolation Forest. *2008 Eighth IEEE International* (2008). http://ieeexplore.ieee.org/xpls/abs

[15] O Lyudchik. 2016. Outlier detection using autoencoders. (2016). http://cds.cern.ch/record/2209085

[16] Yunlong Ma, Peng Zhang, Yanan Cao, and Li Guo. 2013. Parallel auto-encoder for efficient outlier detection. In *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*. 15–17. DOI:http://dx.doi.org/10.1109/BigData.2013.6691791

[17] Lingheng Meng, Shifei Ding, and Yu Xue. 2016. Research on denoising sparse autoencoder. *International Journal of Machine Learning and Cybernetics* (2016), 1–11. DOI:http://dx.doi.org/10.1007/s13042-016-0550-y

[18] Sofia Mosci, Lorenzo Rosasco, Matteo Santoro, Alessandro Verri, and Silvia Villa. 2010. Solving structured sparsity regularization with proximal methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 418–433.

[19] Ulric Neisser. 2004. Pattern Recognition. *Cognitive psychology: Key readings*. (2004). DOI: http://dx.doi.org/10.1016/j.patcog.2011.03.019 arXiv:arXiv:1011.1669v3

[20] Randy C. Paffenroth, Philip C. Du Toit, Louis L Scharf, Anura P Jayasumana, Vidarshana Bandara, and Ryan Nong. 2012. Space-time signal processing for distributed pattern detection in sensor networks. *Proceedings of SPIE - The International Society for Optical Engineering* 8393 (2012), The Society of Photo–Optical Instrumentation Engin. DOI: http://dx.doi.org/10.1117/12.919711

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.

[22] Yu Qi, Yueming Wang, Xiaoxiang Zheng, and Zhaohui Wu. 2014. Robust feature learning by stacked autoencoder with maximum correntropy criterion. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 6716–6720. DOI: http://dx.doi.org/10.1109/ICASSP.2014.6854900

[23] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 833–840.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (oct 1986), 533–536. DOI: http://dx.doi.org/10.1038/323533a0 arXiv:arXiv:1011.1669v3

[25] Craig A. Stewart, George Turner, Matthew Vaughn, Niall I. Gaffney, Timothy M. Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, and Steven Tuecke. 2015. Jetstream: A self-provisioned, scalable science and engineering cloud environment. *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15* (2015), 1–8. DOI: http://dx.doi.org/10.1145/2792745.2792774

[26] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688

[27] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkens-Diehr. 2014. XSEDE: Accelerating scientific discovery. *Computing in Science and Engineering* 16, 5 (sep 2014), 62–74. DOI: http://dx.doi.org/10.1109/MCSE.2014.80 arXiv:http://dx.doi.org/10.1109/MCSE.2014.80

[28] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning - ICML '08* (2008), 1096–1103. DOI: http://dx.doi.org/10.1145/1390156.1390294 arXiv:arXiv:1412.6550v4

[29] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. 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.

[30] J Xie, L Xu, and E Chen. 2012. Image denoising and inpainting with deep neural networks. *Advances in Neural Information Processing* (2012), 1–9. http://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networkshttp://papers.nips.cc/paper/4686-image-denoising

[31] Dan Zhao, Baolong Guo, Jinfu Wu, Weikang Ning, and Yunyi Yan. 2015. Robust feature learning by improved auto-encoder from non-Gaussian noised images. In *IST 2015 - 2015 IEEE International Conference on Imaging Systems and Techniques, Proceedings*. DOI: http://dx.doi.org/10.1109/IST.2015.7294537