


# Anomaly Detection with Robust Deep Autoencoders

*Alessandro Trenta*

- 1 Introduction and Background 
- 2 Robust Deep Autoencoders
- 3 RDAE training
- 4 Results

# Introduction

- Anomaly detection is a fundamental problem when dealing with large amounts of data.
- e.g. Identify with precision which samples or products are anomalous w.r.t. the rest.
- e.g. We want to know if a battery is in good state and to detect strange behaviours by looking at its historical data.
- The main challenge is related to the fact that anomalies are sparse points hidden in a large amount of normal data.
- This presentation is about the Robust Deep Autoencoder models [ZP17] and its applications focusing on anomaly detection.

# Deep Autoencoders

- A Deep Autoencoder (DAE) has two main components: an encoder  $E$  and a Decoder  $D$ . It produces a low dimensional representation of data  $Z = E(X)$ .
- The DAE learns the identity map so that the reconstruction  $\bar{X} = D(E(X))$  is as close as possible to the original input  $X$ .
- $E, D$  can be any mapping between the data space and the coded space. Usually we use FCNN or more complex models (e.g. LSTM or GRU).
- The loss function is

$$\min_{\theta, \phi} \|X - D_{\theta}(E_{\phi}(X))\|_2 \quad (1)$$

# Principal Component Analysis (PCA)

- Assume following data shape:  $N$  samples of  $d$  dimensional data  $X \in \mathbb{R}^{N \times d}$  and w.l.o.g. each feature has 0 mean.
- Mathematically, PCA is an orthogonal linear transformation  $U$  s.t. in the new coordinate system the  $i$ -th component has the  $i$ -th greatest data variance.
- PCA can be used for dimensionality reduction: project the data on the first  $k < d$  principal components.
- How to find the new basis:

$$w_1 = \arg \max_{\|w\|_2=1} \|Xw\|_2^2 = \arg \max_w \frac{w^T X^T X w}{w^T w} \quad (2)$$

# Robust Principal Component Analysis

- Robust Principal Component Analysis (RPCA) is a generalization of PCA that aims to reduce the sensitivity of PCA to outliers and noise.
- Idea: separate noise and outliers, then learn a low dimensional representation of cleaned data.
- Assume that data  $X$  can be represented as  $X = L + S$ :  $L$  has low rank and is the low-dimensional representation of  $X$  while  $S$  is a sparse matrix containing outliers and anomalous data.

# Robust Principal Component Analysis

- The problem can be addressed as:

$$\min_{L,S} \rho(L) + \lambda \|S\|_0 \quad (3)$$

$$\text{s. t. } X = L + S \quad (4)$$

where  $\rho(\cdot)$  is the rank of a matrix and we used the zero norm.  
 $\lambda$  controls the sparsity.

- This optimization problem is NP-hard and tractable only for small matrices.
- New objective:

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1 \quad (5)$$

where  $\|\cdot\|_*$  is the nuclear norm i. e. the sum of singular values of a matrix.

# Robust Deep Autoencoders

- Robust Deep Autoencoders (RDAE) idea: combine the representation learning of DAEs and the anomaly detection capability of RPCA.
- Leave out the sparse part  $S$  and learn a low-dimensional representation of  $L$ .
- There are two kinds of RDAE, one for  $l_1$  regularization and one for  $l_{2,1}$ .



# RDAE and regularization

- We try to decompose data as  $X = L_D + S$  as in RPCA and combine the losses in this optimization problem:

$$\min_{\theta} \|L_D - D_{\theta}(E_{\theta}(L_D))\|_2 + \lambda \|S\|_1 \quad (6)$$

$$\text{s.t. } X = L_D + S \quad (7)$$

- The parameter  $\lambda$  controls the sparsity of  $S$ . A smaller  $\lambda$  means that the norm of  $S$  is less important w.r.t. DAE loss (better reconstruction, less outliers) and viceversa.
- Finding the best value for  $\lambda$  is the main challenge.

## Regularization and anomalies

- The RDAE with  $l_1$  penalization assumes that outliers and noise are not structured. The  $l_1$  penalty just induces sparsity. We could have different kind of anomalies:
- Feature (column) wise: a feature is corrupted in many samples e.g. a broken pixel in a sensor.
- Data (row) wise: a particular sample is anomalous and very different from the majority of them.

# The $l_{2,1}$ norm

- The  $l_{2,1}$  norm for feature anomalies is defined as ( $X \in \mathbb{R}^{N \times d}$ ):

$$\|X\|_{2,1} = \sum_{j=1}^d \|X_j\|_2 = \sum_{j=1}^d \left( \sum_{i=1}^N |X_{ij}|^2 \right)^{\frac{1}{2}} \quad (8)$$

- We can also do the other way around: to recognize data anomalies (by row) just apply the  $l_{2,1}$  norm to  $X^T$ .
- The final optimization problem for the RDAE with  $l_{2,1}$  regularization for data anomalies is then

$$\min_{\theta} \|L_D - D_{\theta}(E_{\theta}(L_D))\|_2 + \lambda \|S^T\|_{2,1} \quad (9)$$

$$\text{s.t. } X - L_D - S = 0 \quad (10)$$

# The proximal operator

- To see in detail the training procedure for the RDAE we first need to consider the proximal operator.
- General framework: find the solution to  $\min f(x) + \lambda g(x)$  where  $g$  is convex. Consider

$$\text{prox}_{\lambda, g}(x) = \arg \min_y g(y) + \frac{1}{2\lambda} \|x - y\|_2^2 \quad (11)$$

- In the case of proximal gradient optimization the iterative step is defined as:

$$x^{k+1} = \text{prox}_{\lambda, g}(x^k - \alpha \nabla f(x^k)) \quad (12)$$

- For  $g(x) = \|x\|_1$ , the solution to the proximal problem is

$$\text{prox}_{\lambda, l_1}(x) = \begin{cases} x_i - \lambda, & x_i > \lambda \\ x_i + \lambda, & x_i < -\lambda \\ 0, & x_i \in [-\lambda, \lambda] \end{cases} \quad (13)$$

for  $S \in \mathbb{R}^{N \times d}$  it gets applied element by element.

- $g(x) = \|x\|_{2,1}$ : for feature anomalies we obtain (letting  $S_{\cdot j}$  be the column vector  $S_{ij}, j = 1, \dots, N$ )

$$(\text{prox}_{\lambda, l_{2,1}}(S))_{ij} = \begin{cases} S_{ij} - \lambda \frac{S_{ij}}{\|S_{\cdot j}\|_2}, & \|S_{\cdot j}\|_2 > \lambda \\ 0, & \|S_{\cdot j}\|_2 \leq \lambda \end{cases} \quad (14)$$

The proposed optimization method is a two step iterative process (ADMM).

Given input  $X \in \mathbb{R}^{N \times d}$ , initialize  $L_D \in \mathbb{R}^{N \times d}$ ,  $S \in \mathbb{R}^{N \times d}$  as zero matrices,  $L_S = X$  and initialize the DAE randomly. For each iteration do:

- $L_D = X - S$
- Minimize  $\|L_D - D_\theta(E_\theta(L_D))\|_2$  with backpropagation (DAE).
- Set  $L_D = D(E(L_D))$  as the reconstruction.
- Set  $S = X - L_D$ .
- Optimize  $S$  using a  $\text{prox}_{\lambda, f}$  function of choice.
- If  $c_1 = \frac{\|X - L_D - S\|_2}{\|X\|_2} < \epsilon$  or  $c_2 = \frac{\|L_S - L_D - S\|_2}{\|X\|_2} < \epsilon$  we have early convergence.
- Set  $L_S = L_D + S$ .

Return  $L_D$  and  $S$ .

# Results

- We now have a look at how the model performs on some tasks.
- We will initially use the MNIST digit dataset, using the 50000 training images available.
- Data was flattened from images of shape  $(28, 28, 1)$  into vectors of length 784. Train data is then a matrix in  $\mathbb{R}^{50000 \times 784}$ .
- Pixel values are converted from integers between 0 and 255 to floats between 0 and 1.

# $l_1$ Robust Deep Autoencoder

- We take a RDAE with a FCNN architecture with layers of size 784 (input), 200 and 10 (the bottleneck and hidden feature layer). 10 outer iterations and 100 inner iterations.
- The training images get corrupted with a percentage of pixel (from 5% to 50%) changed to a random value between 0 and 1. These are used to train the RDAE.
- Implemented using python 3.8 and Tensorflow 2.9.



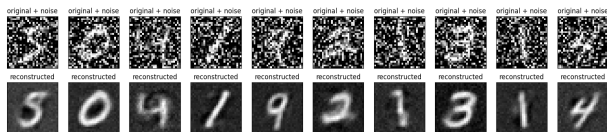


Figure: RAE cleaned data, corruption 50%

## $l_{2,1}$ Robust Deep Autoencoder

- To assess the performance of RDAE in anomaly detection we start by using a synthetic labeled dataset.
- All the 4 digit images for training are collected in our dataset.
- Then, some images are chosen at random from all the other digits until they reach 5% of total images in the dataset.
- These will be considered as the outliers of our data.

- The  $l_{2,1}$  RDAE is trained on this dataset without any side information. Same architecture as before.
- The only parameter that requires tuning is  $\lambda$ . We assess performance with the accuracy, precision, recall and  $F_1 = 2 \frac{PR}{P+R}$  metrics.
- We consider an instance anomalous whenever the  $S$  matrix has non-zero entries on its row.

## Anomaly detection performance

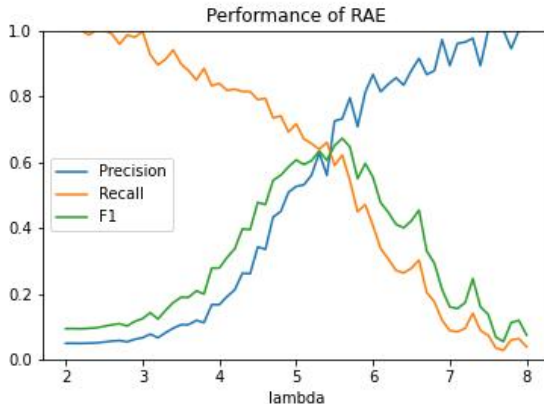


Figure:  $L_{2,1}$  RDAE anomaly detection performance.  $\lambda$  from 2 to 8

- The maximum performance is obtained with  $\lambda = 5.468$  with an  $F_1$  score of 0.668.
- From  $\lambda = 5$  to  $\lambda = 6$  the RDAE has a  $F_1$  score almost everytime above 0.55.
- Precision and recall are both in the range  $[0.6, 0.7]$  for  $\lambda$  close to the optimal.
- Let's look at the reconstructions by the DAE of the RDAE  $D(E(X))$ , the cleaned images  $L_D$  and the sparse image  $S$  from 3 different values of  $\lambda$ : low, optimal and high.

## Original Images data

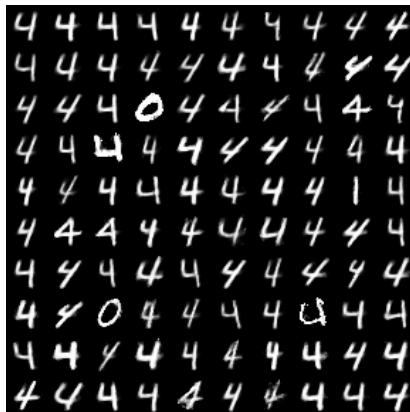
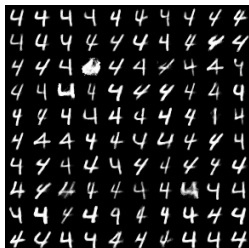
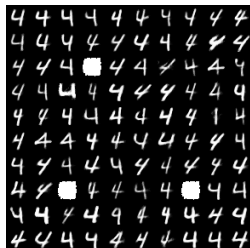


Figure: Original images for the  $L_{2,1}$  RDAE

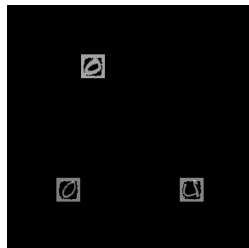
$$\lambda = 5.468$$



(a)  $\bar{X}$ , reconstruction



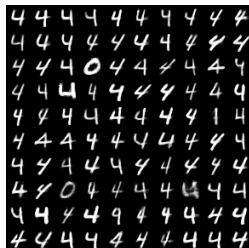
(b)  $L_D$ , cleaned data



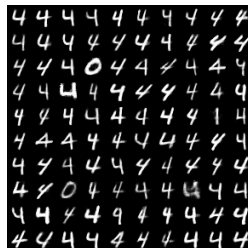
(c)  $S$ , outliers

Figure: Accuracy: 0.970, precision: 0.722, recall: 0.621, F1 score: 0.668

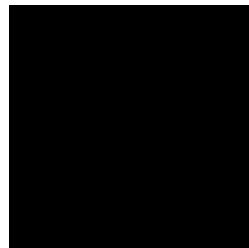
$\lambda = 8.0$



(a)  $\bar{X}$ , reconstruction



(b)  $L_D$ , cleaned data

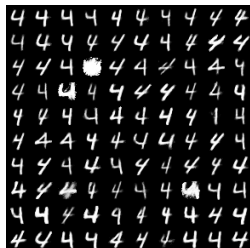


(c)  $S$ , outliers

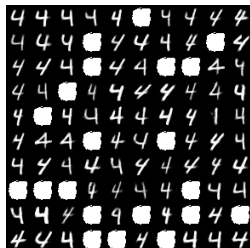
Figure: Accuracy: 0.953, precision: 1.00, recall: 0.0386, F1 score: 0.0743



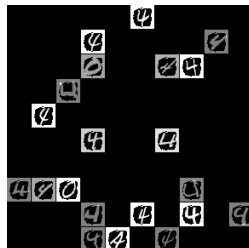
$$\lambda = 4.0$$



(a)  $\bar{X}$ , reconstruction



(b)  $L_D$ , cleaned data



(c)  $S$ , outliers

Figure: Accuracy: 0.788, precision: 0.167, recall: 0.839, F1 score: 0.278

- The performance of the RDAE as outlier detector is compared with the one obtained using the isolation forest method.
- The isolation forest method is based on the idea that outliers are few and very different from the rest.
- These outliers get recognized using isolation trees which try to separate points from others.
- The only parameter to be optimized is the outlier fraction (from 0 to 0.5).

## Isolation forest performance

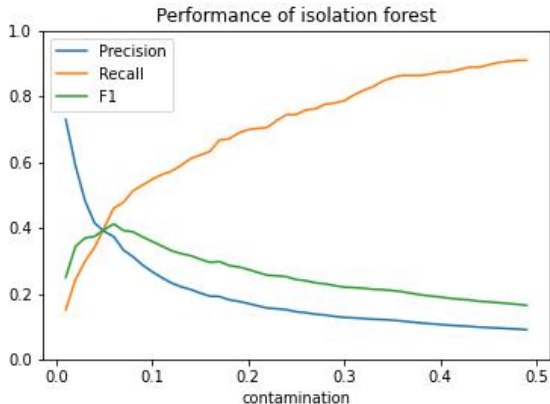


Figure: The performance is far worse than RDAE.

## Real World Experiment

- Now we have assessed the performance on a synthetic dataset we want to see how to apply this model to real world challenges.
- Data is in general UNLABELED. Specially with loads of data, we can't do hand labeling of anomalies.
- We want to see if the model still works for time series.
- This could have a true application in Electric Vehicles products: given historical data from sensor, detect if batteries had strange behaviour.



## Time series experiment

- In this case we are going to use a dataset similar to our interest.
- It is taken from the Numenta Anomaly Benchmark (NAB). The database is called machine temperature system failure.
- It is the sensor data of an internal component of a large, industrial machine. It should have 3 kinds of anomalies: mainly planned shutdowns of the machine, then the second kind should be difficult to detect and directly led to the third anomaly, a catastrophic failure of the machine.

# Data processing pipeline

- Data has 22464 timesteps in total. Data signals are taken every 5 minutes.
- I chose to consider subsequences of length 144: this corresponds of windows of 12 hours. The final dataset has then 22321 training time series.
- Data is normalized all together to be in  $(0, 1)$ .

## RDAE architectures and analysis

I tried using 3 architectures for the autoencoder part in the RDAE.

- The first one is a Dense Neural Network with hidden layers of 60 and 20. It is trained for 20 outer iterations and 50 inner iterations for the autoencoder, batch size 256 ,  $\epsilon = 10^{-8}$ .
- The second and the third one are a LSTM and a GRU with two layers of 32 and 16 units, 10 outer iterations and 25 inner iterations with same batch size as before.
- Since data is unlabeled we don't have a clear benchmark for the value of  $\lambda$ . I tried different values to see how the number of outliers scales.

# Anomalies found

$\lambda$	0.1	0.5	0.7	1.0	2.0	3.0	3.2	3.3	4
<b>Dense</b>	All	751	250	14	0	0	0	0	0
<b>LSTM</b>	All	7525	4208	2068	306	109	74	9	0
<b>GRU</b>	All	7277	4505	2454	331	139	103	80	0

**Table:** Anomalies found by the two architectures w.r.t.  $\lambda$



## Dense RDAE

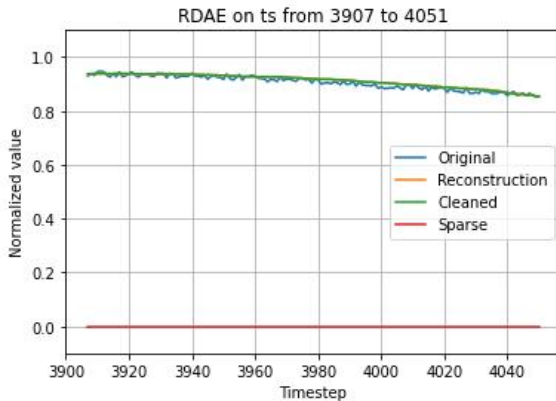


Figure: Example of a non anomaly subsequence for  $\lambda = 1.0$

## Dense RDAE

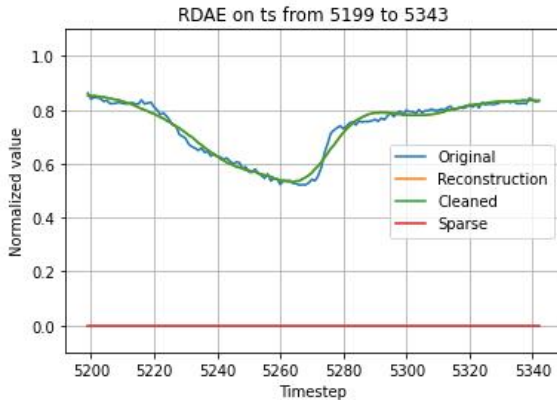


Figure: Example of a non anomaly subsequence for  $\lambda = 1.0$

## Dense RDAE

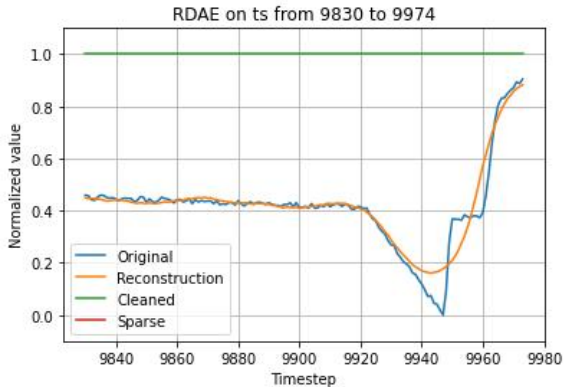


Figure: Example of a anomaly subsequence for  $\lambda = 1.0$

## Dense RDAE

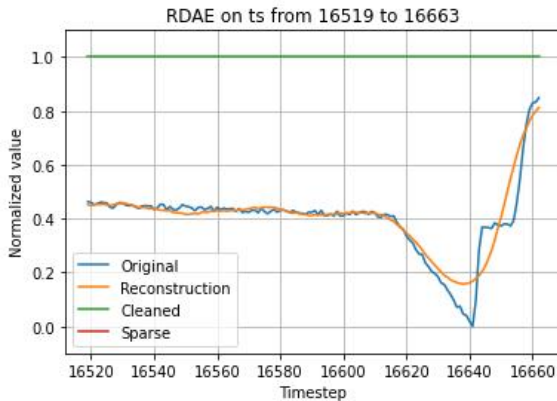


Figure: Example of a anomaly subsequence for  $\lambda = 1.0$

## Dense RDAE

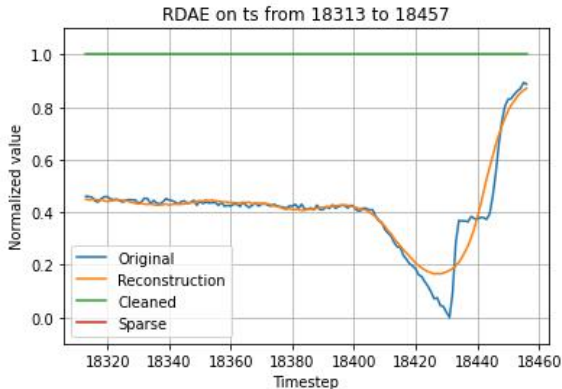


Figure: Example of a anomaly subsequence for  $\lambda = 1.0$

## Dense RDAE

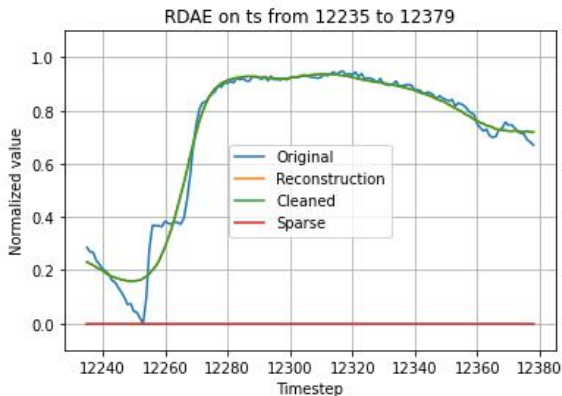


Figure: Example of a non anomaly subsequence for  $\lambda = 1.0$

## Dense RDAE

- All of the anomalies found reach the 0 value (min temperature of all time series).
- Note that the different anomalies found DO NOT overlap. So each of the failures is only recognized once.
- This may also create problems, since as you can see one failure is not recognized as anomaly.
- In general, the reconstruction is a non-noisy version of the signal.

# LSTM RDAE

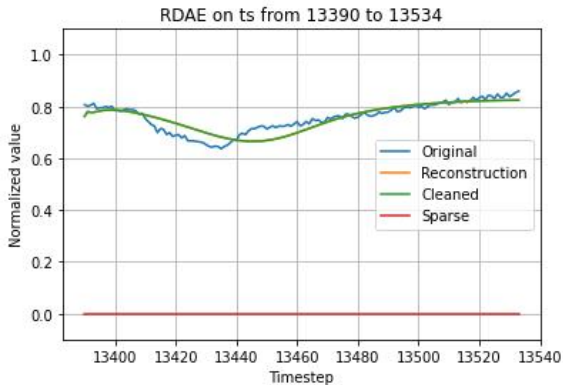


Figure: Example of a non anomaly subsequence for  $\lambda = 3.3$



# LSTM RDAE

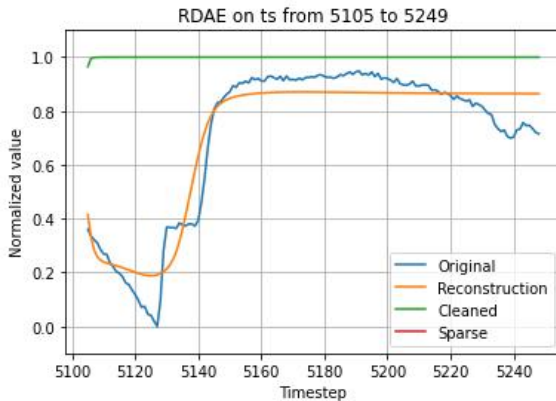
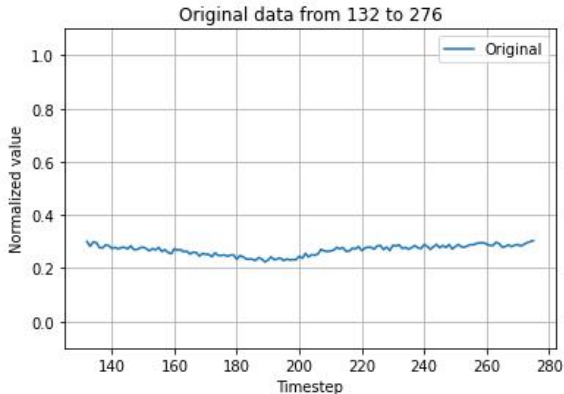


Figure: Example of a anomaly subsequence for  $\lambda = 3.3$

# Isolation Forest

- I tried to fit an isolation forest on the same data, to see if we get the same kind of outliers.
- With all outlier fraction values tested (0.0001, 0.0005, 0.001, 0.01), non of them showed significant results.
- On the contrary, almost all the anomalies detected I saw were normal time series, almost flat.

## Isolation forest anomaly example



**Figure:** Example of an anomaly found by the isolation forest with outlier fraction of 0.001

## Final comments

- The RDAE is a powerful tool for denoising and anomaly detection with many application both for images, time series and can detect different kinds of anomalies in a general setting.
- Unfortunately the main quest is to find the correct  $\lambda$  value. With unlabeled data this could be very difficult.
- In that case, a possible way out is to know the approximate anomaly rate and to hope the anomalies found match the true ones.

- Note an important thing: after we train the model there is no way to find anomalies on new given data.
- The  $L_D$  and  $S$  matrices are produced only in the training procedure.
- We can still denoise images with the Autoencoder part.
- It could be tried to add new data after some iteration, without re-initializing. This requires training again the autoencoder, which is the high computational part.

<https://github.com/AlexThirty/SaMLMfTSA>

Thank you!



Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha.

Unsupervised real-time anomaly detection for streaming data.  
*Neurocomputing*, 262:134–147, 2017.  
Online Real-Time Learning Strategies for Data Streams.



Emmanuel J. Candes, Xiaodong Li, Yi Ma, and John Wright.  
Robust principal component analysis?, 2009.



Neal Parikh.  
Proximal algorithms.  
*Foundations and Trends in Optimization*, 1:127–239, 01 2014.



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*, page 665–674, 2017.