# Anomaly Detection with Robust Deep Autoencoders

## original article by C. Zhou and R. C. Peffenroth

*Alessandro Trenta*

Scuola Normale Superiore

# Deep Autoencoders

- A Deep Autoencoder (DAE) is constituted by 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 (LSTM or GRU).
- The loss function: it is the minimum reconstruction error w.r.t. some parametrized encoding and decoding functions and a distance (in this case the $L_2$ norm)

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

# Principal Component Analysis

- Assume following data shape: $N$ samples of $d$ dimensional data $X \in \mathbb{R}^{N \times d}$ and assume each feature has 0 mean.
- Principal Component Analysis (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.
- Ideally we want to fit a $n$-ellipsoid into the data. The length of an axis of the ellipsoid represents the variance of data along that axis.
- PCA is often used for dimensionality reduction or encoding: we can project the data on the first $k < d$ principal components.

# Principal Component Analysis

Mathematically we can define:

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

for the first component. Then for the $k$-th component we first subtract the first $k-1$ principal component from $X$

$$\hat{X}_k = X - \sum_{i=1}^{k-1} X w_i w_i^T \tag{3}$$

and finally solving again the similar problem:

$$w_k = \arg\max_{\|w\|_2=1} \|\hat{X}_k w\|_2^2 = \arg\max_w \frac{w^T \hat{X}_k^T \hat{X}_k w}{w^T w} \tag{4}$$

# Robust Principal Component Analysis

- Robust Principal Component Analysis (RPCA) is a generalization of PCA that aims to reduce the sensitivity of PCA to outliers.
- The idea is to find a low-dimensional representation of data cleaned from the sparse outliers.
- 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:

$$\min_{L,S} \rho(L) + \lambda \|S\|_0 \tag{5}$$

$$\text{s. t. } \|X - L - S\|_F^2 = 0 \tag{6}$$

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

- This optimization problem is NP-hard and tractable only for small metrices.
- The used objective is instead:

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1 \tag{7}$$

$$\text{s. t. } \|X - L - S\|_F^2 = 0 \tag{8}$$

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

# Robust Deep Autoencoders

- Robust Deep Autoencoders (RDAE) combine the representation learning of DAEs and the anomaly detection capability of RPCA.
- Noise and outliers are incompressible in the lower dimensional space we want to represent our data in.
- We want to exclude anomalies and learn a low dimensional representation of data.
- We will see two RDAE typed, one for $l_1$ regularization and one for $l_{2,1}$.

# RDAE with $l_1$ regularization

- We try to decompose data as $X = L_D + S$ as in RPCA.
- By removing the noise $S$ the autoencoder can better reconstruct $L_D$.
- We then combine the two losses in the following minimization problem

$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_0 \tag{9}$$

$$\text{s.t. } X - L_D - S = 0 \tag{10}$$

- The parameter $\lambda$ controls the sparsity of $S$.

# The role of $\lambda$

- A smaller $\lambda$ means that the norm of $S$ is less important w.r.t. the DAE loss.
- The model will reconstruct better but recognize less outliers. Helpful if we want a more faithful representation.
- A larger $\lambda$, instead, gives more importance to the norm of $S$ as a loss.
- This means that the model will recognize more (or even too much) outliers, sacrificing some reconstruction performance.
- Finding the right value for $\lambda$ is the main challenge for this model.

# The true objective

- The previous loss is higly non tractable. We focus on the following problem:

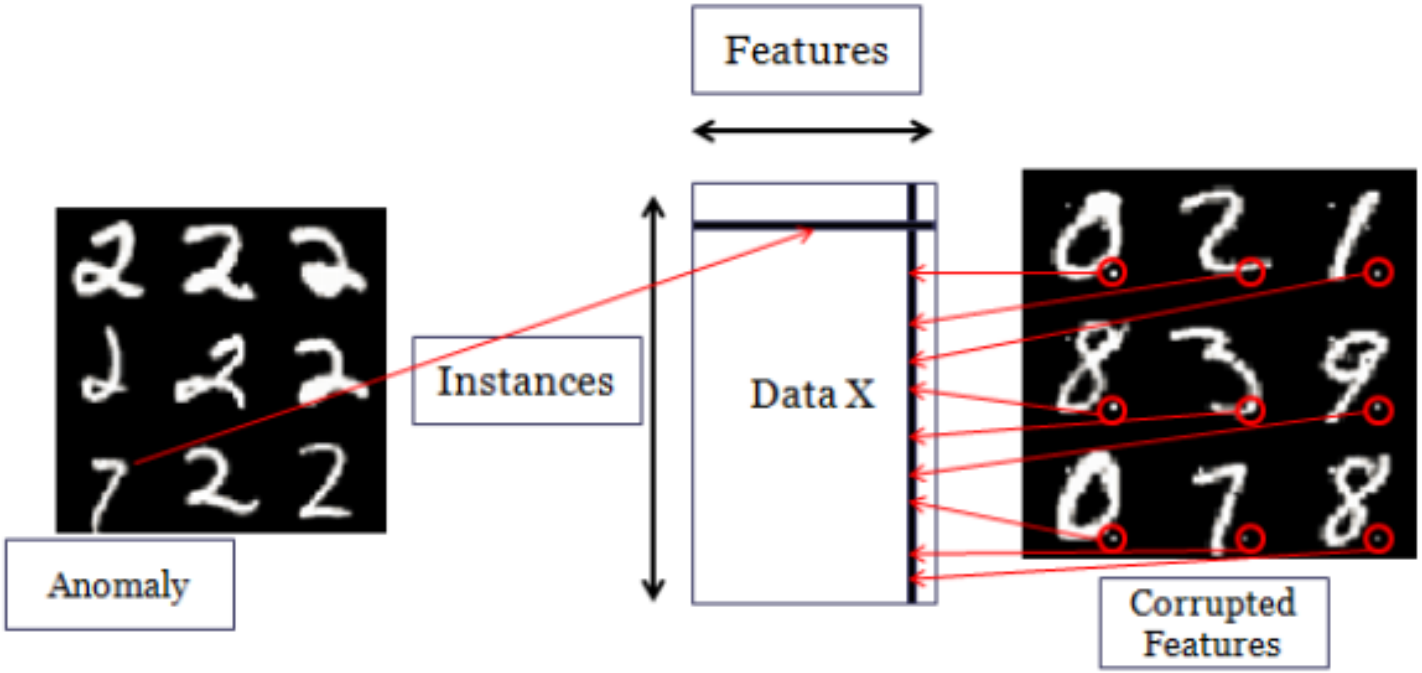
$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_1 \tag{11}$$

$$\text{s.t. } X - L_D - S = 0 \tag{12}$$

- The autoencoder is trained with $L_D$, the noise and outliers free part.
- There is no specific requirement about the DAE mappings.

# Regularization

- 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.
- Data (row) wise: a particular sample is anomalous.


Features
Instances
Data X
Anomaly
Corrupted
Features

# The $l_{2,1}$ norm

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

$$\|X\|_{2,1} = \sum_{j=1}^{n} \|X_j\|_2 = \sum_{j=1}^{n} \left( \sum_{i=1}^{N} |X_{ij}|^2 \right)^{\frac{1}{2}} \tag{13}$$

- The $l_{2,1}$ norm can be seen as introducting a $l_2$ norm regularization over data for each feature and then adding a $l_1$ regularization accross features.
- 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

$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S^T\|_{2,1} \tag{14}$$

$$\text{s.t. } X - L_D - S = 0 \tag{15}$$

- For detecting feature anomalies we just need to change the objective to

$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S\|_{2,1} \tag{16}$$

$$\text{s.t. } X - L_D - S = 0 \tag{17}$$

# 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

$$\text{prox}_{\lambda,g}(x) = \arg\min_y g(y) + \frac{1}{2\lambda}\|x - y\|_2^2 \tag{18}$$

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

$$x^{k+1} = \text{prox}_{\lambda,g}(x)(x^k - \alpha\nabla f(x^k)) \tag{19}$$

- In this case we then want to obtain a solution of the problems

$$\text{prox}_{\lambda, l_1}(x) = \arg\min_y l_1(y) + \frac{1}{2\lambda}\|x - y\|_2^2 \tag{20}$$

$$\text{prox}_{\lambda, l_{2,1}}(x) = \arg\min_y l_{2,1}(y) + \frac{1}{2\lambda}\|x - y\|_2^2 \tag{21}$$

- For the $l_1$ norm, the solution to the proximal problem is

$$\text{prox}_{\lambda,l_1}(x) = \begin{cases} x_i - \lambda, & x_i > \lambda \\ x_i + \lambda, & x_i < -\lambda \\ 0, & x_i \in [-\lambda, \lambda] \end{cases} \tag{22}$$

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

- $l_{2,1}$ norm: for feature anomalies we obtain (letting $S_{\cdot j}$ be the column vector $S_{ij}, j = 1, \ldots, N$)

$$(\text{prox}_{\lambda,l_{2,1}}(S))_{ij} = \begin{cases} S_{ij} - \lambda \frac{S_{ij}}{\|S_{\cdot j}\|_2}, & \|S_{\cdot j}\|_2 > \lambda \\ 0, & \|S_{\cdot j}\|_2 \leq \lambda \end{cases} \tag{23}$$

- Substitute $S$ with $S^T$ for data anomalies.

# The main algorithm

- The method used to train the RDAE is the Alternating Direction Method of Multipliers (ADMM).
- It is a two-step iterative process to solve the problem

$$\min_{\theta} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda\|S^T\|_{2,1} \tag{24}$$

$$\text{s.t. } X - L_D - S = 0 \tag{25}$$

- First, we fix $S$ and optimize the DAE loss $\|L_D - D_\theta(E_\theta(L_D))\|_2$ with backpropagation.
- Then, we fix $L_D$ and optimize the regularization term with the proximal method.

The full procedure is the following: given input $X \in \mathbb{R}^{N \times n}$, initialize $L_D \in \mathbb{R}^{N \times n}, S \in \mathbb{R}^{N \times n}$ 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.
- Set $L_D = D(E(L_D))$ as the reconstruction.
- Set $S = X - L_D$.
- Optimize $S$ using a $\text{prox}_{\lambda,l.}$ function of choice.
- If $c_1 = \frac{\|X - L_D - S\|_2}{\|X\|_2} < \epsilon$ or $c_2 = \frac{\|LS - L_D - S\|_2}{\|X\|_2} < \epsilon$ we have early convergence.
- Set $L_S = L_D + S$.

Return $L_D$ and $S$.

# Results

- I tried to reproduce some of the results by the original article.
- We will initially use the MNIST digit daraset, which consists of 50000 train samples amd 10000 test images.
- 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 walues are converted from integers between 0 and 255 to floats between 0 and 1.

# Implementation

- The RDAE and the standard DAEs used in this experimental tries were implemented using Tensorflow 2.9.1 on python 3.8.
- For the random forest classifier and the isolation forest models were taken from SciKit-learn version 1.1.1.
- Full implementation and details can be found on GitHub

# $l_1$ Robust Deep Autoencoder

- To assess the performance of the $l_1$ RDAE the proposed procedure is the following:
- The training images get corrupted with a percentage of pixel (from 5% to 50%) changed to a random value between 0 and 1.
- Both the RDAE with $l_1$ regularization and a standard DAE (with same architecture as the DAE from the RDAE) are trained on these corrupted images.
- From the RDAE obtain the two matrices $L_D$, the cleaned data, and $S$, the sparse and anomalous part.

# $l_1$ Robust Deep Autoencoder

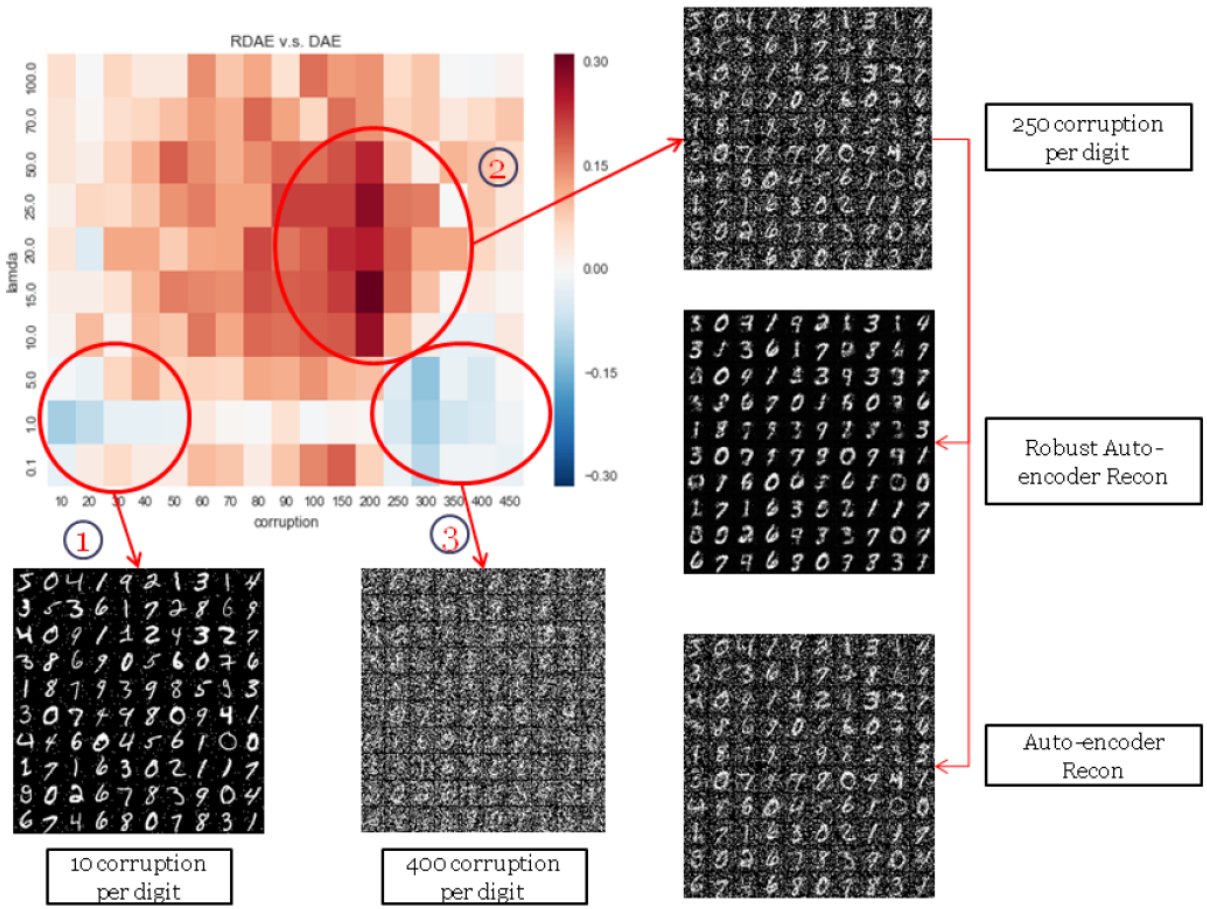
- Given the trained models we obtain the two bottleneck layer encodings: $Z = E(X)$ for the DAE and $Z_R = E_R(L_D)$ for the RDAE.
- These encodings are divided into a training test $(\frac{2}{3})$ and a test set $(\frac{1}{3})$.
- Two random forest classifiers are then trained on the training encodings extracted.
- We test how these RF classifiers perform on the test set.

- This should show how well the model is able to extract the important features in the low dimensional representation.
- In this case the RDAE and DAE need to denoise the images and summarize the features.
- Both architectures are simple FCNN with layers of size 784 (input), 200 and 10 (the bottleneck and hidden feature layer).
- The RDAE was trained for 10 outer iterations with 100 inner iterations each, while the DAE was trained for 100 epochs. The batch size is 256.

# $L_1$ RDAE analysis

- Unfortunately, I could not replicate the results by the article, even by changing parameters.
- The authors showed a significant improvement in performance by the RDAE, which didn't happen in my experiments.
- We see the results in function of $\lambda$ and the corruption percentage.

# Original results



# Deep Autoencoder RF performance

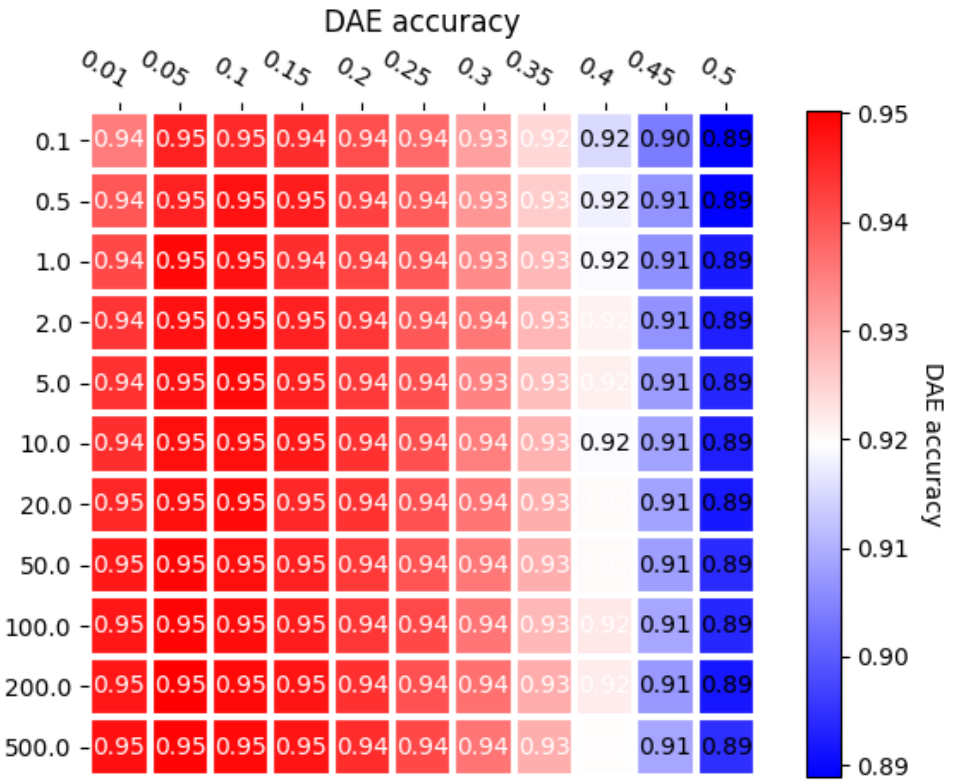



Figure: Performance of the Random forest on the hidden layer of base Deep Autoencoder on different $\lambda$, corruption

# $l_1$ RDAE RF performance

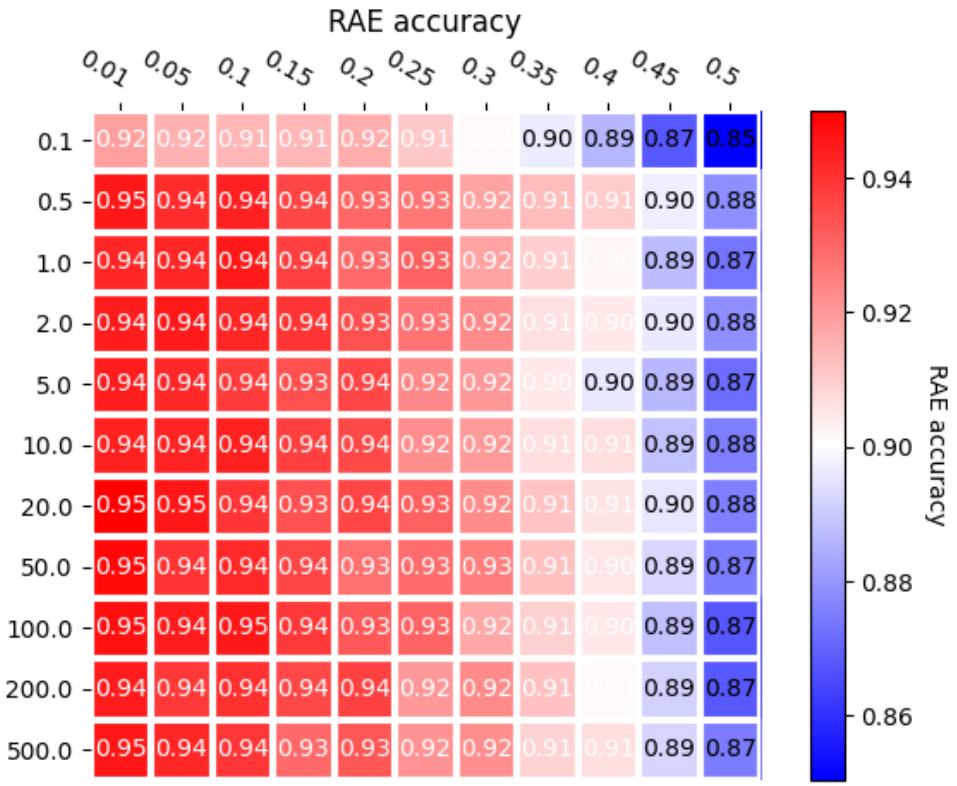



Figure: Performance of the Random forest on the hidden layer of base $l_1$ RDAE on different $\lambda$, corruption

# Performance comparison

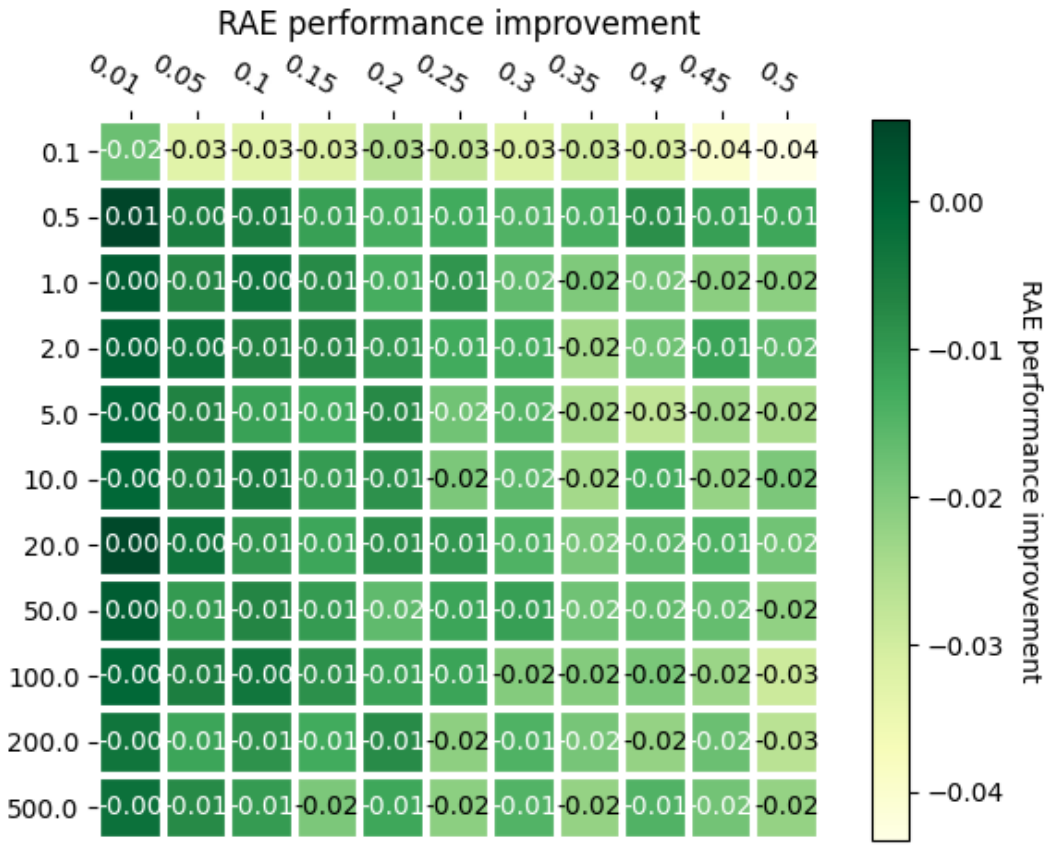



Figure: Performance increase of using RDAE on different $\lambda$, corruption

# Comments on performance

- In almost each case the RDAE performed slightly worse than the DAE.
- The RDAE performance is not bad, but the simpler DAE approach seems to work better at this task.
- The 30% improvement they showed is impossibile to get from this DAE performance.
- The best value for $\lambda$ is 20.0, even though almost all values showed similar results.

# Reconstruction/cleaning of noisy images

- It is really fascinating and interesting to see the denoising capabilities of the two models.
- I selected $\lambda = 20.0$ as it is the value for which latent representation of data gave approximately the best results for every percentage of corruption.
- For the RDAE we consider the cleaned $L_D$ version obtaining from the training procedure.
- For the DAE we consider the reconstruction $\bar{X} = D(E(X))$

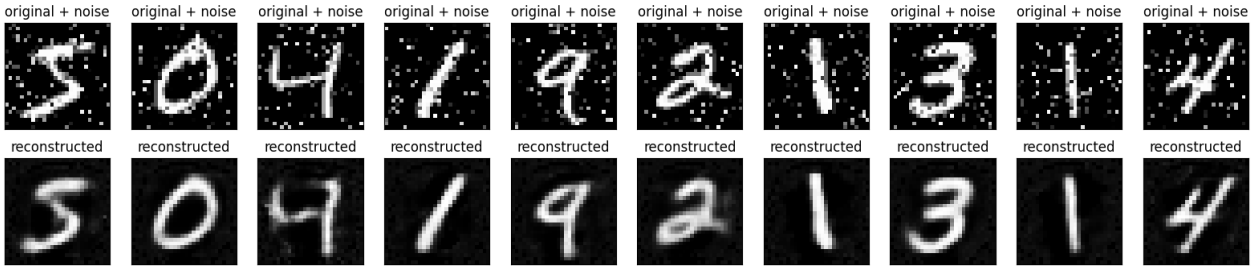



Figure: DAE cleaned data, corruption 10%

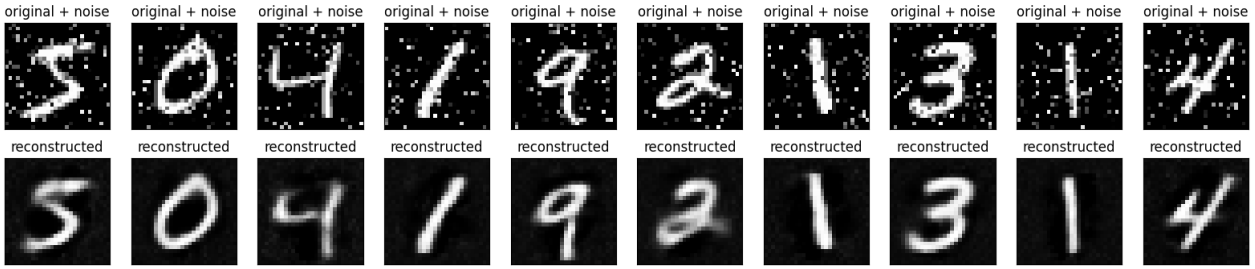



Figure: RAE cleaned data, corruption 10%

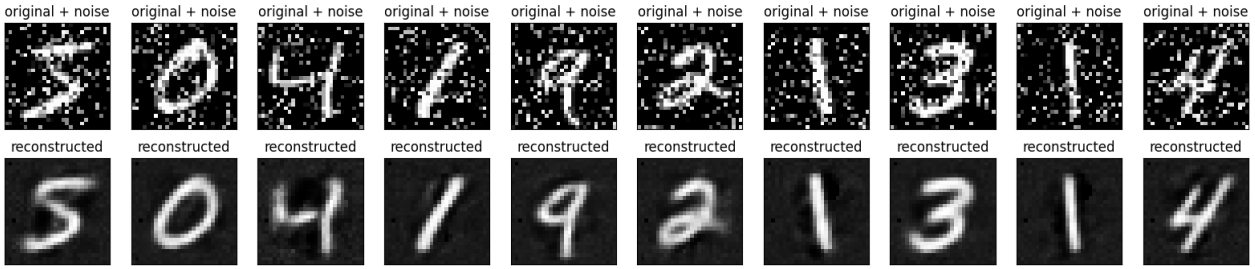



Figure: DAE cleaned data, corruption 20%

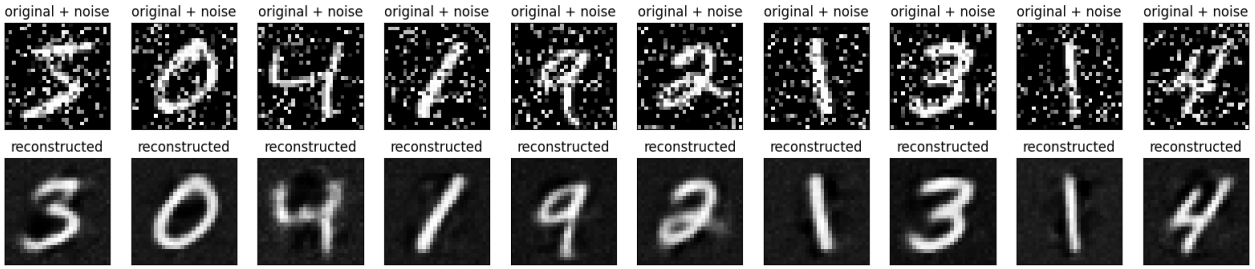



Figure: RAE cleaned data, corruption 20%

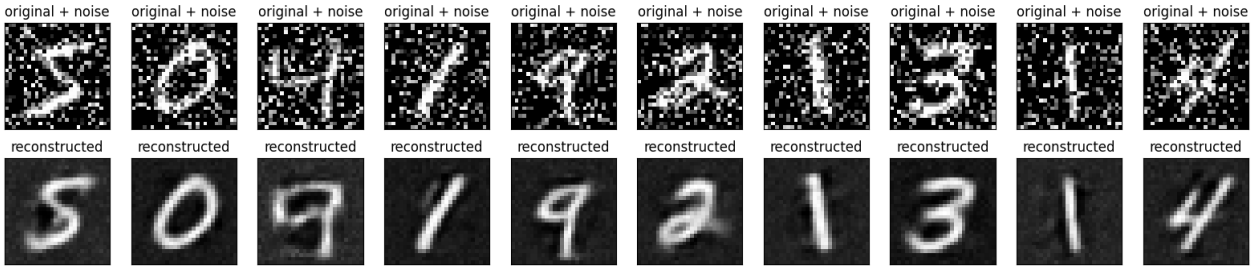



Figure: DAE cleaned data, corruption 30%

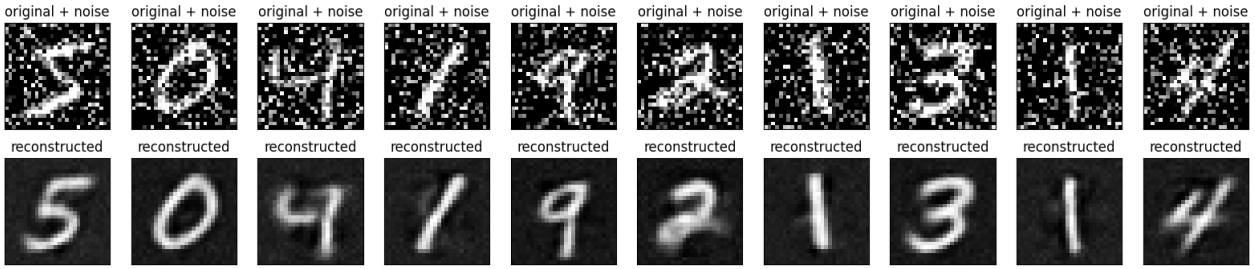



Figure: RAE cleaned data, corruption 30%

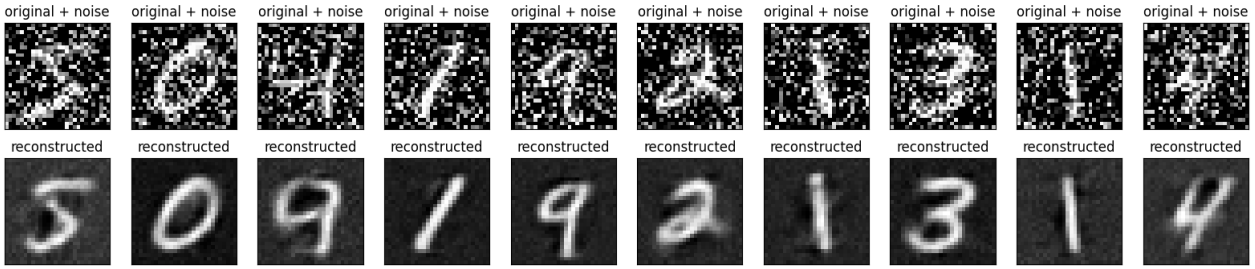



Figure: DAE cleaned data, corruption 40%

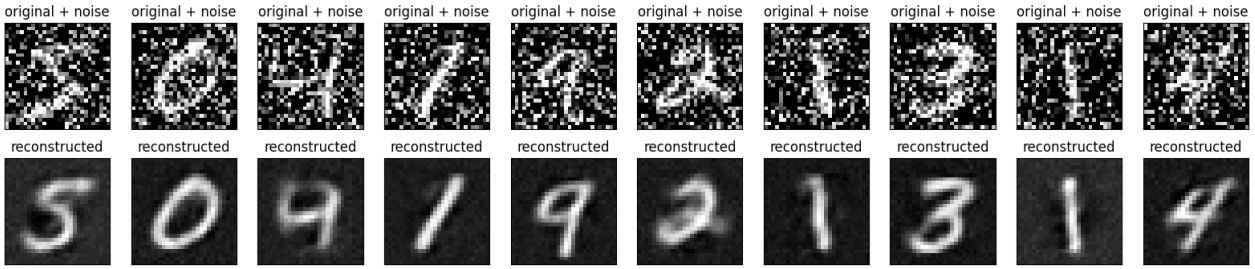



Figure: RAE cleaned data, corruption 40%

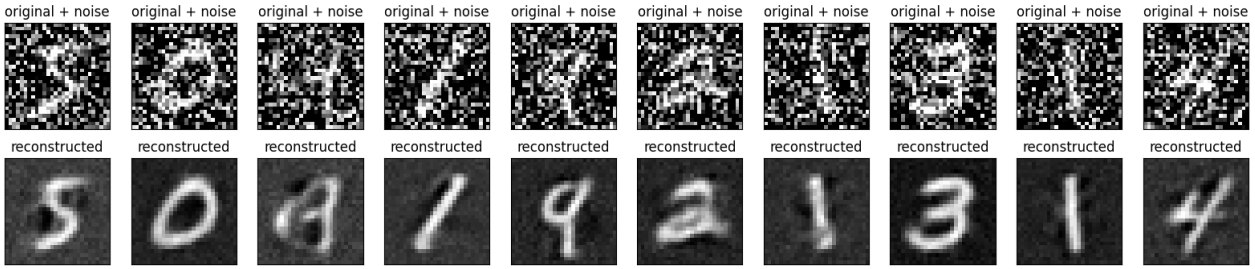



Figure: DAE cleaned data, corruption 50%

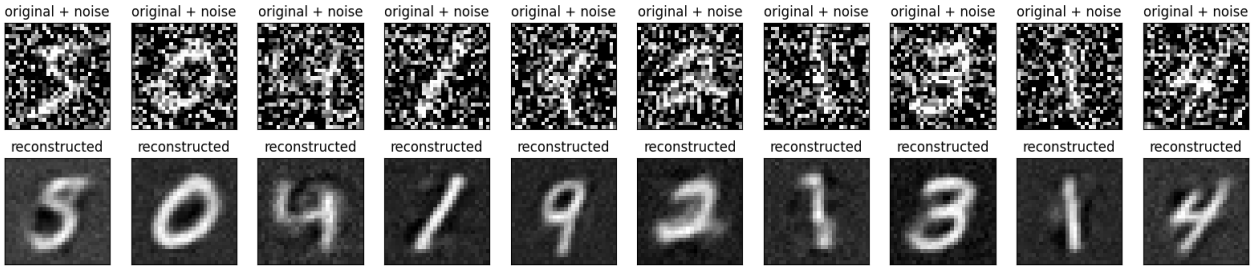



Figure: RAE cleaned data, corruption 50%

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

- The data anomaly detection experiment aims to separeate a specific digit of the MNIST dataset from others.
- All the 4 digit images in the training set 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. It has to recognize outliers completely on its own.
- The model architecture is the same as for the $l_1$ RDAE experiment.
- The only parameter that requires tuning is $\lambda$.

- Model performance is assessed by how well it recognizes the correct "outliers".
- The metrics used are the accuracy, the precision score, the recall score and the F1 score defined down below.

$$ACC = \frac{TP + TN}{P + N} \quad P = \frac{TP}{TP + FP} \tag{26}$$

$$R = \frac{TP}{TP + FN} \quad F1 = 2\frac{P \cdot R}{P + R} \tag{27}$$

- The F1 score, which tries to average in some way precision and recall, is the metrics used to select the $\lambda$ parameter.

# Original performance

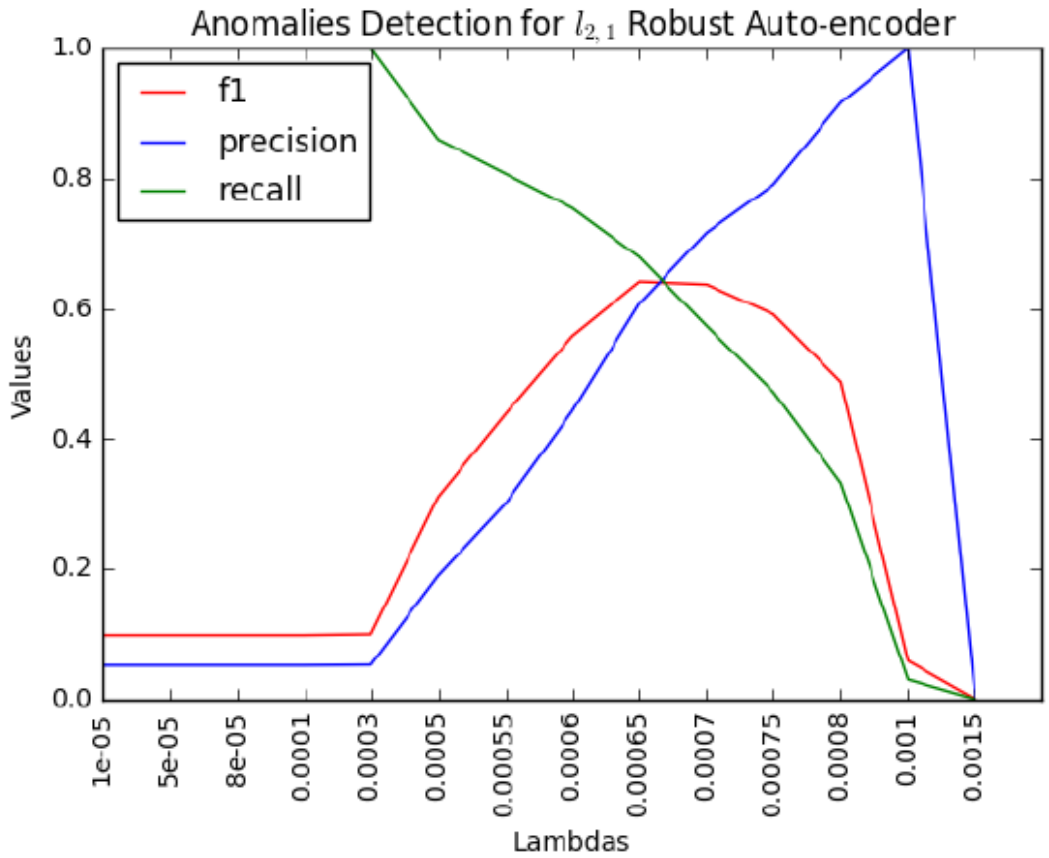



Figure: $L_{2,1}$ RDAE anomaly detection performance from original article

# Anomaly detection performance

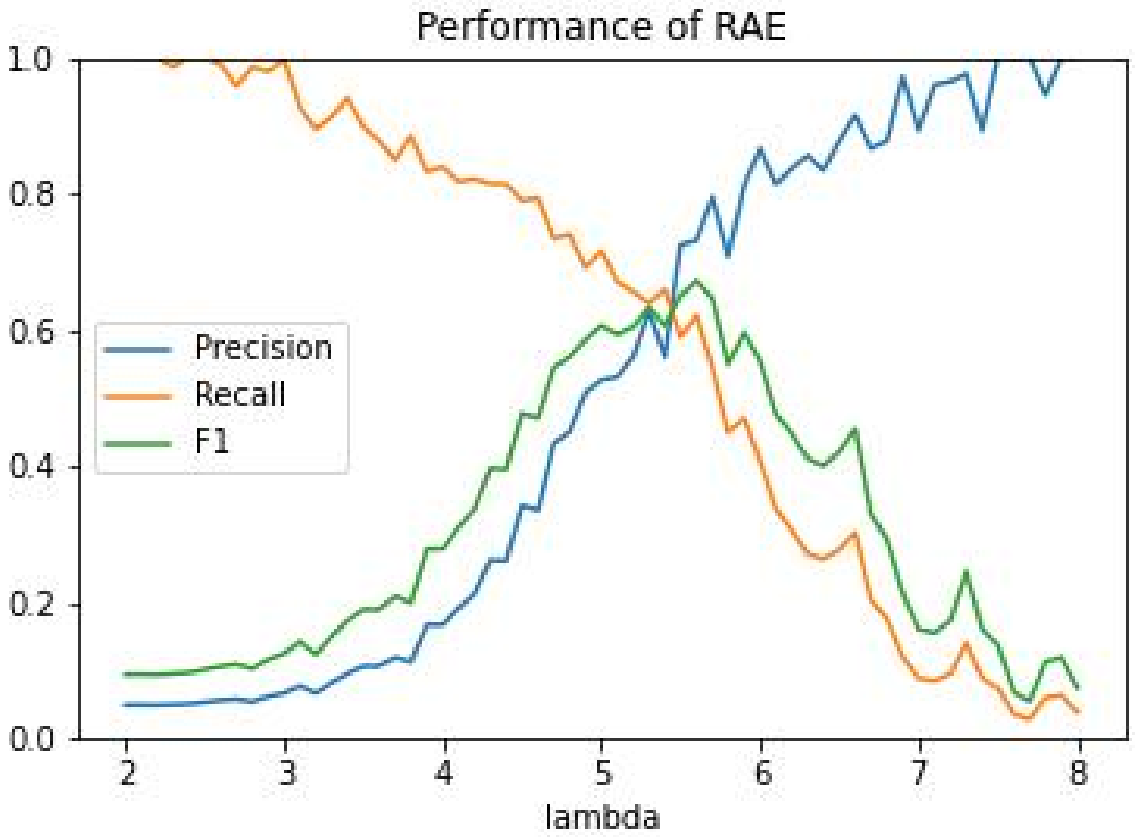



Figure: $L_{2,1}$ RDAE anomaly detection performance. $\lambda$ from 2 to 8

# Anomaly detection performance

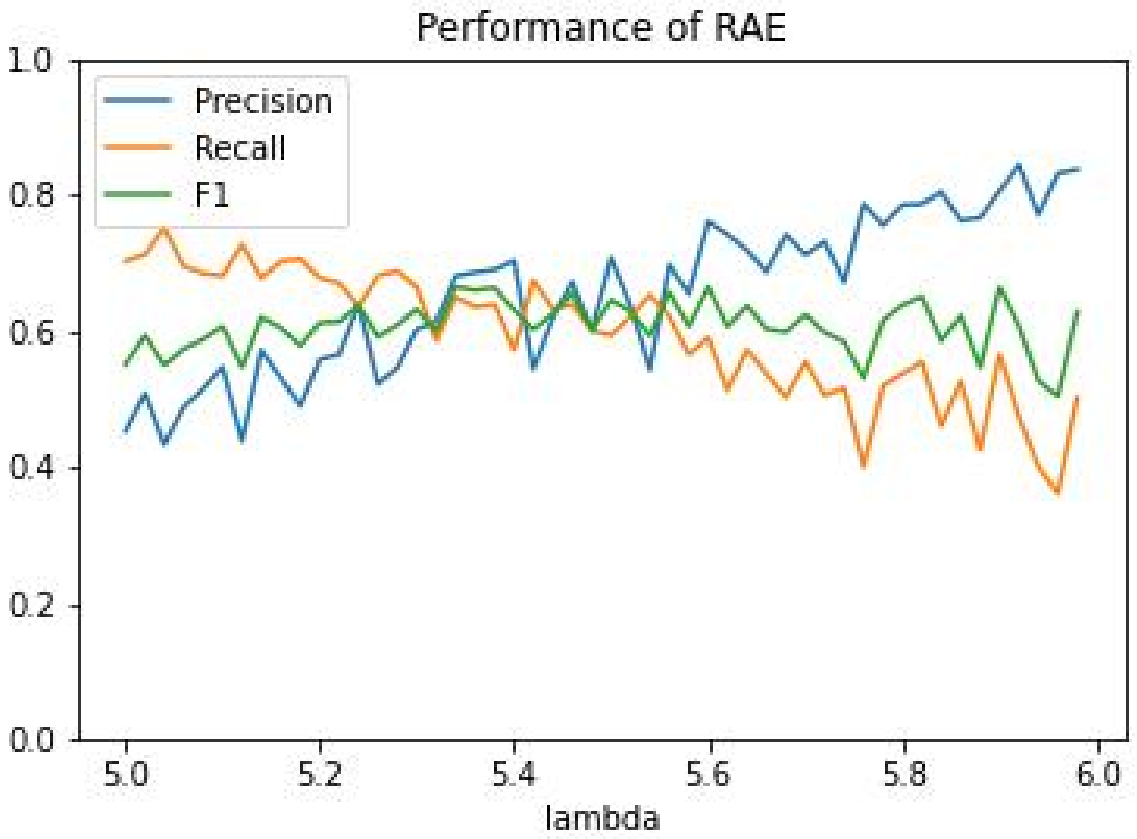



Figure: $L_{2,1}$ RDAE anomaly detection performance. $\lambda$ from 5 to 6

# Anomaly detection performance


Figure: $L_{2,1}$ RDAE anomaly detection performance. $\lambda$ from 5.4 to 5.6

- The maximum performance is obtained with $\lambda = 5.468$ with an $F_1$ score of 0.668.
- Focusing on all values from $\lambda = 5$ to $\lambda = 6$ the RDAE has an accuracy of over 95% in recognizing anomalies. The $F_1$ score in this range is almost everytime above 0.55.
- The $F_1$ score is almost everytime above 0.6 for $\lambda$ in the $[5.4, 5.6]$ range.
- Best result obtained by the authors is an $F_1$ score of 0.64 for $\lambda = 0.00065$. This different value of $\lambda$ is in my opinion due to the different parameters of the neural network.

For each value of $\lambda$ we analize 3 images:

- The reconstruction of the original images from the DAE in the RDAE $\bar{X} = D(E(X))$.
- The final $L_D$ image (the "clean" version, in this case it should only contain 4s)
- $S$ image, which should be non empty only for outliers.

We look 3 different values for $\lambda$: the best one identified above, 8.0 which adds too much penalization with few outliers identified and 4.0 which is a low value and a lot of 4s are considered outliers.

# Original Images data

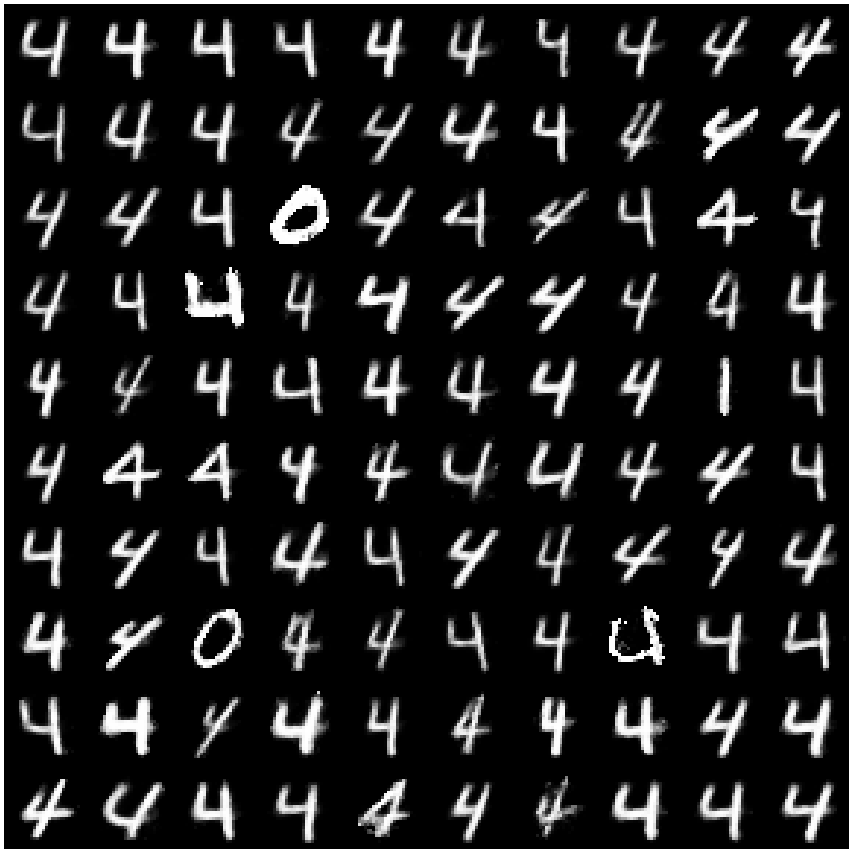
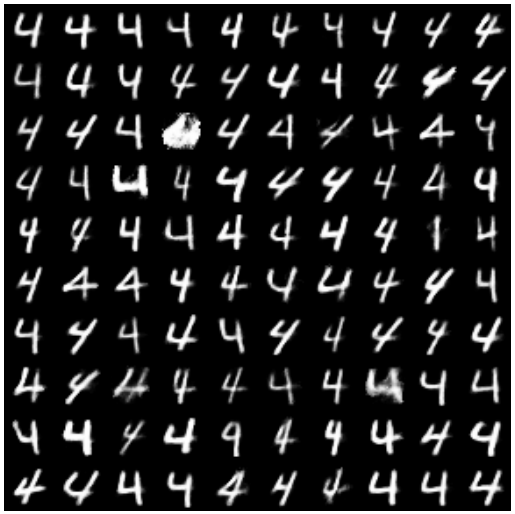
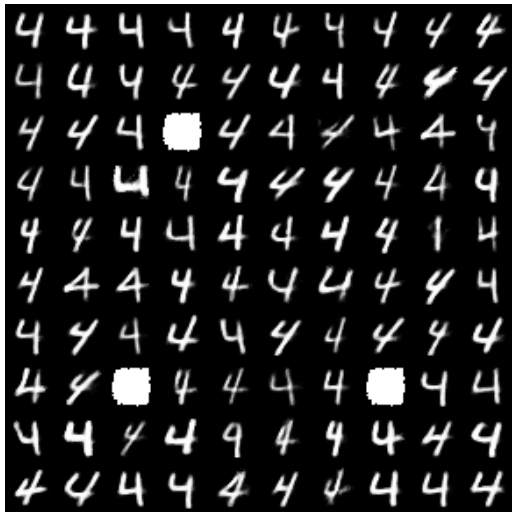


Figure: Original images for the $L_{2,1}$ RDAE

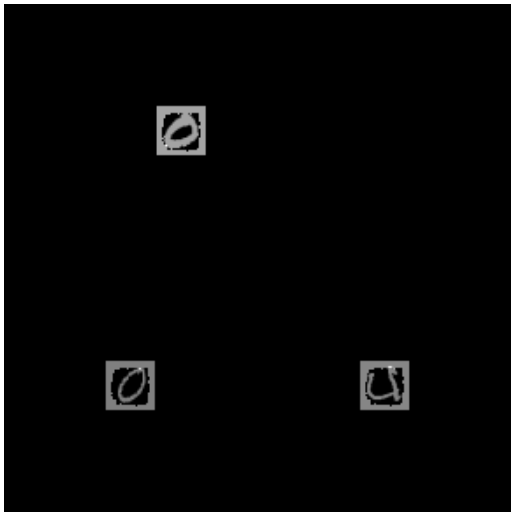
# $\lambda = 5.468$



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



(b) $L_D$, cleaned data



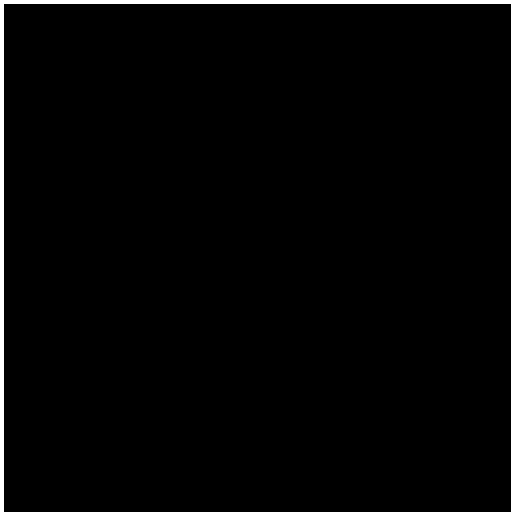
(c) $S$, outliers

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

# $\lambda = 8.0$

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

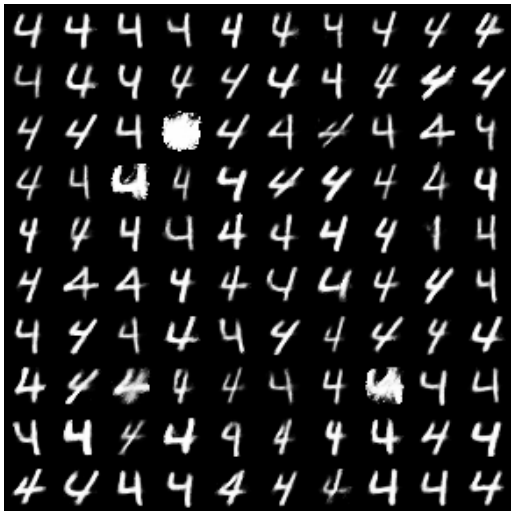
(b) $L_D$, cleaned data



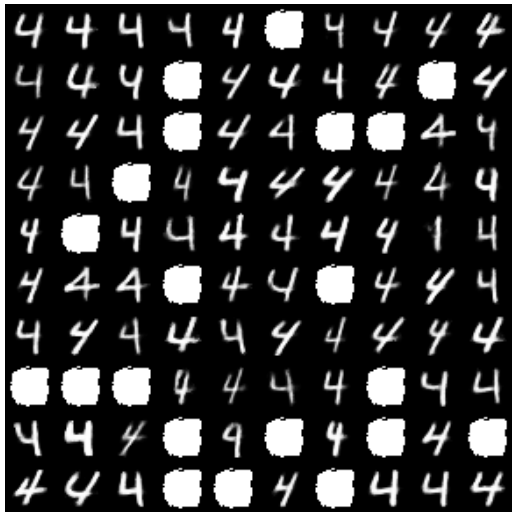
(c) $S$, outliers

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

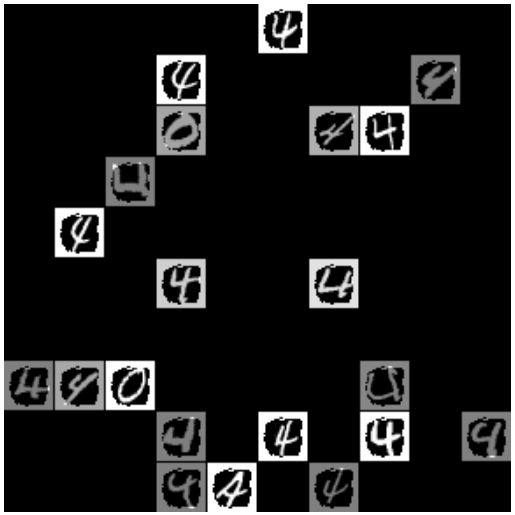
# $\lambda = 4.0$



(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 outlied detector is compared with the one obtained using the isolation forest method.
- The isolation forest method was a SOTA method for outlier detection. It is based on the idea that outliers are few, different and separated from the rest.
- These outliers gets 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

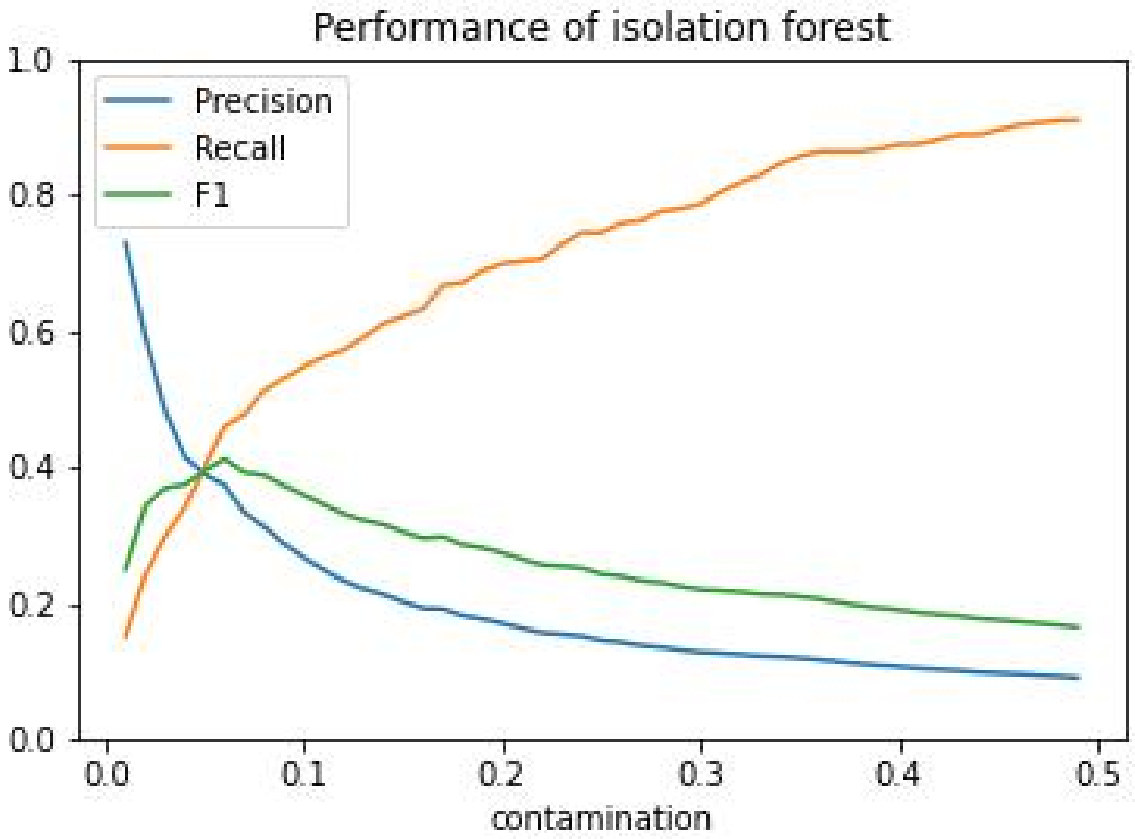



Figure: Isolation forest performance

- The best results is 0.41 with the outlier fraction set to the value of 0.06 (which is really close to the outlier true fraction of 5%).
- This result is really close to the original article. The authors obtained a best $F_1$ score of 0.37 with 0.11 outlier fraction.
- The little difference is in my opinion related to the different datasets.
- In each case the performance by isolation forest is far worse than the RDAE.

# Time series experiment

- I tried to apply this method to time series.
- In this case we are going to use a dataset 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 mahcine. It should have 3 anomalies: the first anomaly is a planned shutdown of the machine. The second anomaly is difficult to detect and directly led to the third anomaly, a catastrophic failure of the machine.
- Data has 22464 timesteps in total. I chose to consider subsequences of length 144. The final dataset has then 22321 training time series.
- Data is normalized all togheter to be in $(0, 1)$.

# RDAE architectures

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.

# Analysis

- Since data is unlabeled we don't have a clear benchmark for finding the correct value for $\lambda$.
- I tried different values for $\lambda$ to see how the number of outliers scales.
- For each architecture I picked some random anomalies and non-anomalies, to show how the RDAE is acting on time series and to have a look at what kind of anomalies it detectes.

# Anomalies found

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

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

# Dense RDAE

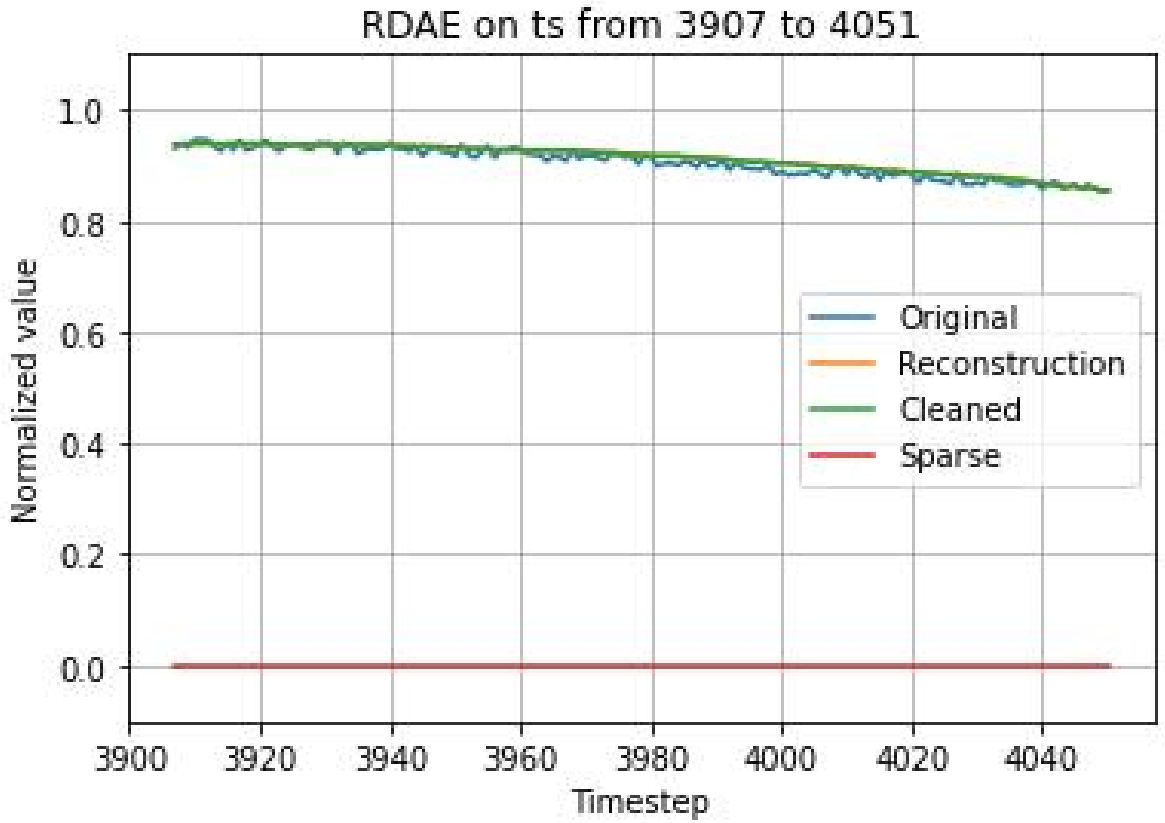



Figure: Example of a non anomaly subsequence for $\lambda = 1.0$

# Dense RDAE

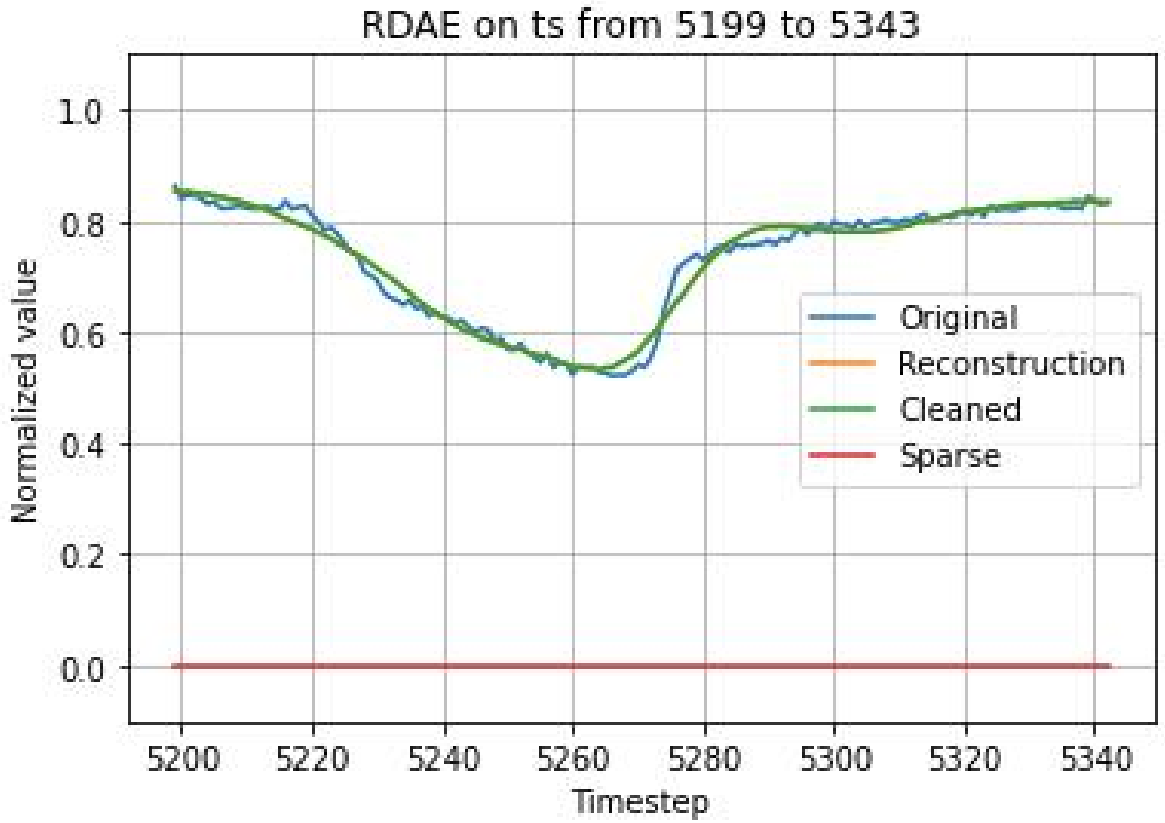



Figure: Example of a non anomaly subsequence for $\lambda = 1.0$

# Dense RDAE

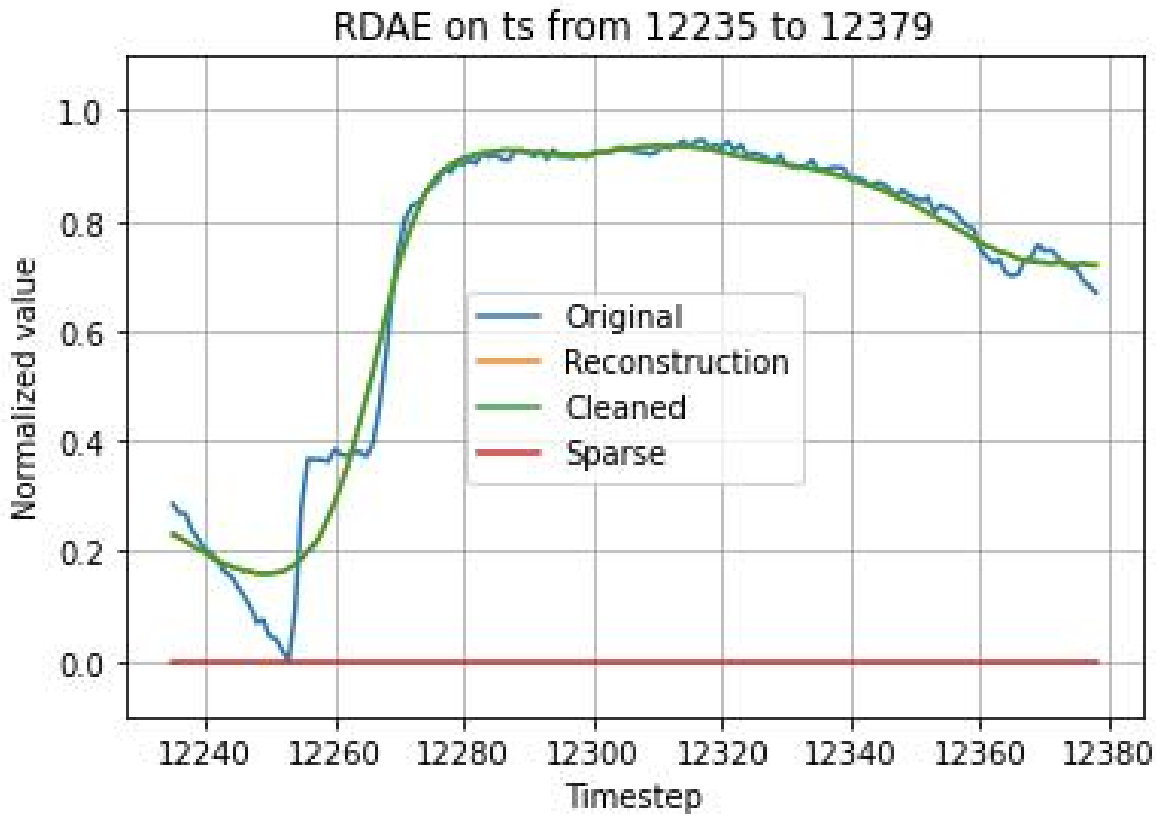



Figure: Example of a non anomaly subsequence for $\lambda = 1.0$

# Dense RDAE

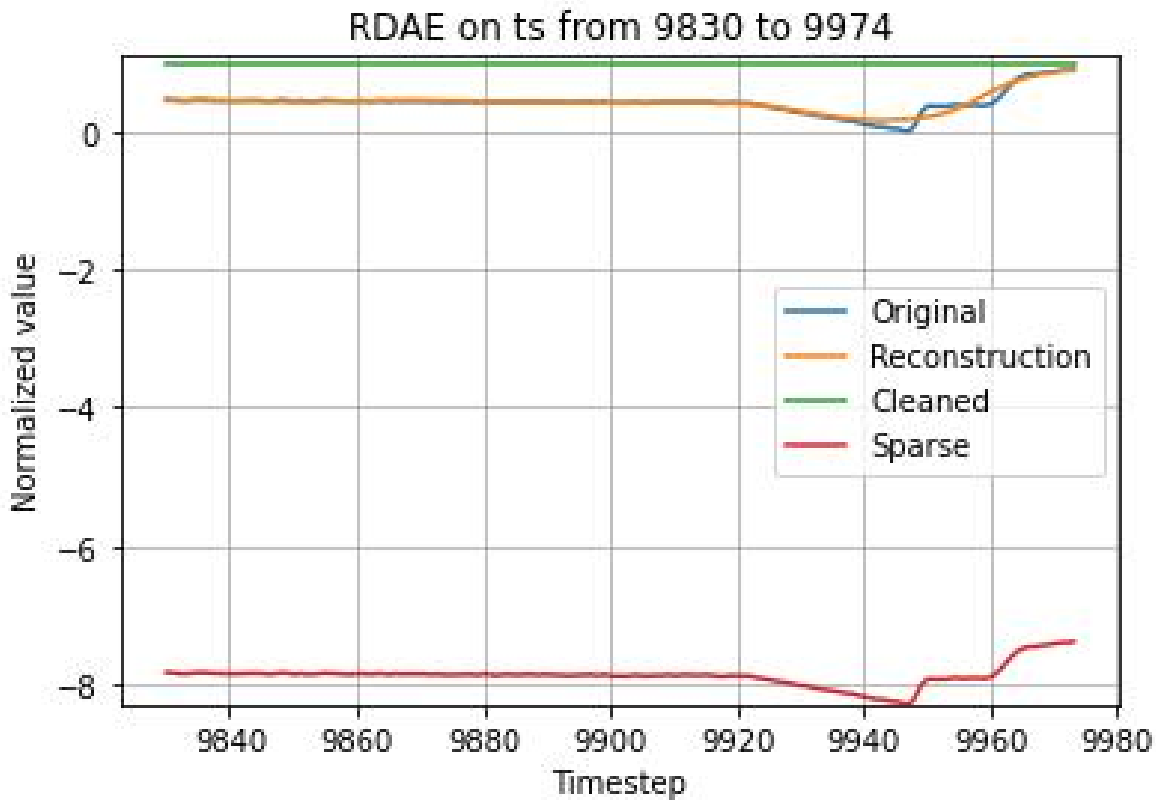



Figure: Example of a anomaly subsequence for $\lambda = 1.0$

# Dense RDAE

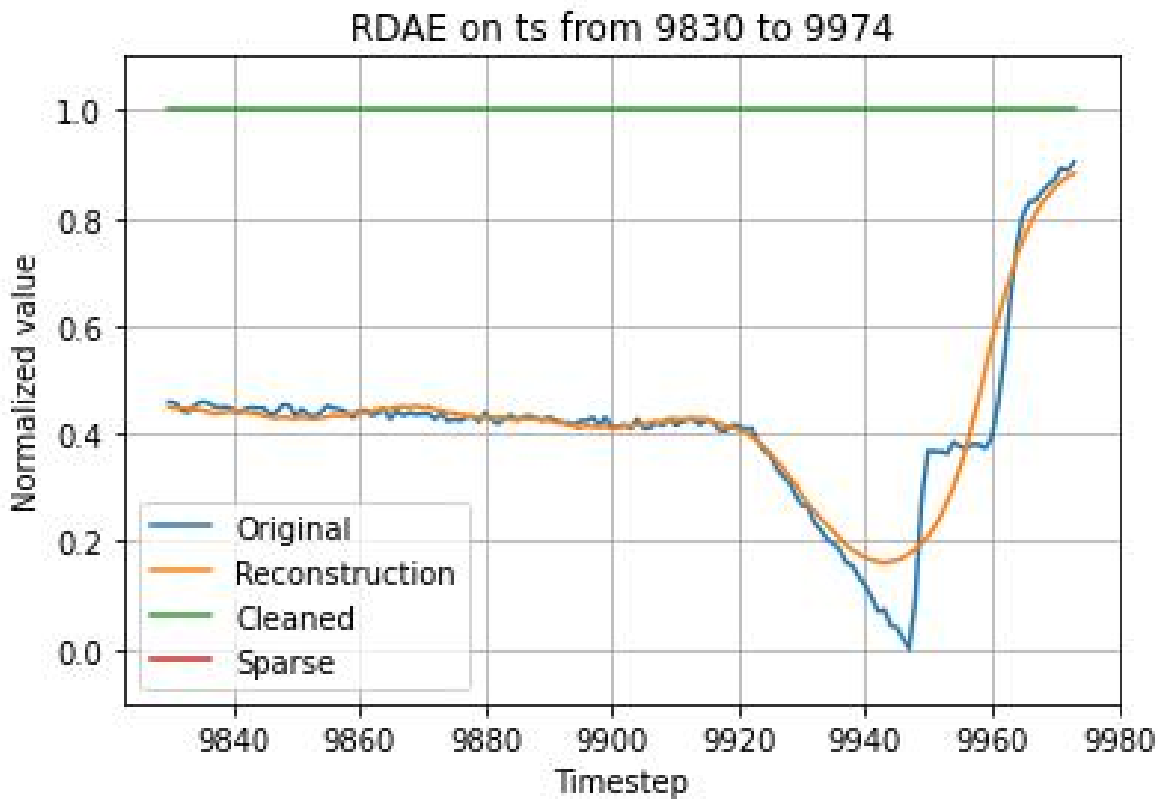



Figure: Example of a anomaly subsequence for $\lambda = 1.0$

# Dense RDAE

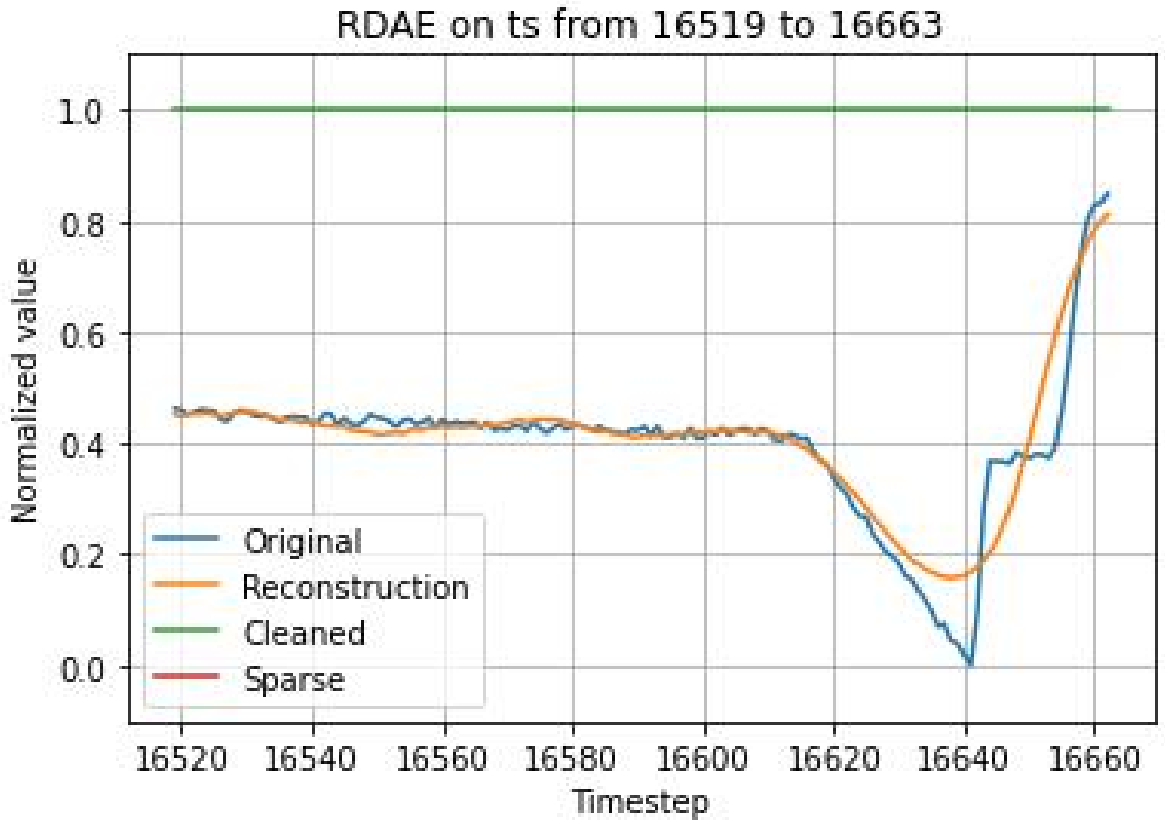



Figure: Example of a anomaly subsequence for $\lambda = 1.0$

# Dense RDAE

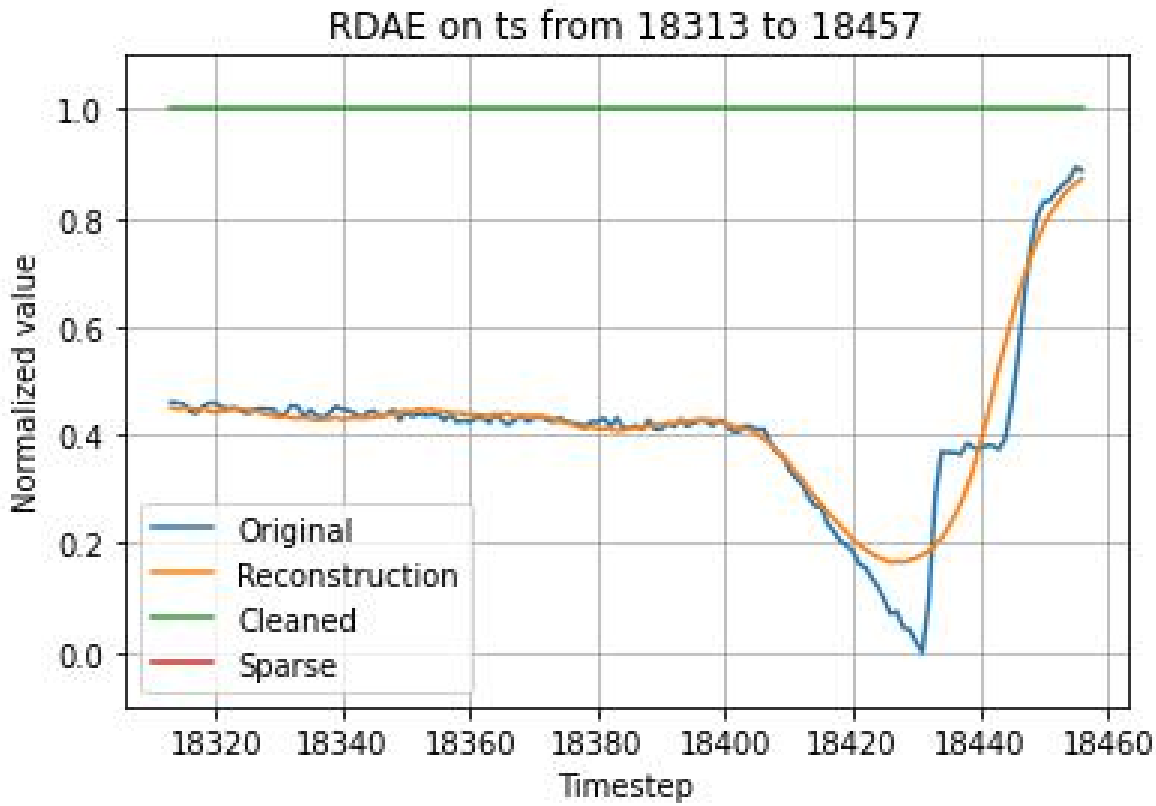



Figure: Example of a anomaly subsequence for $\lambda = 1.0$

# 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

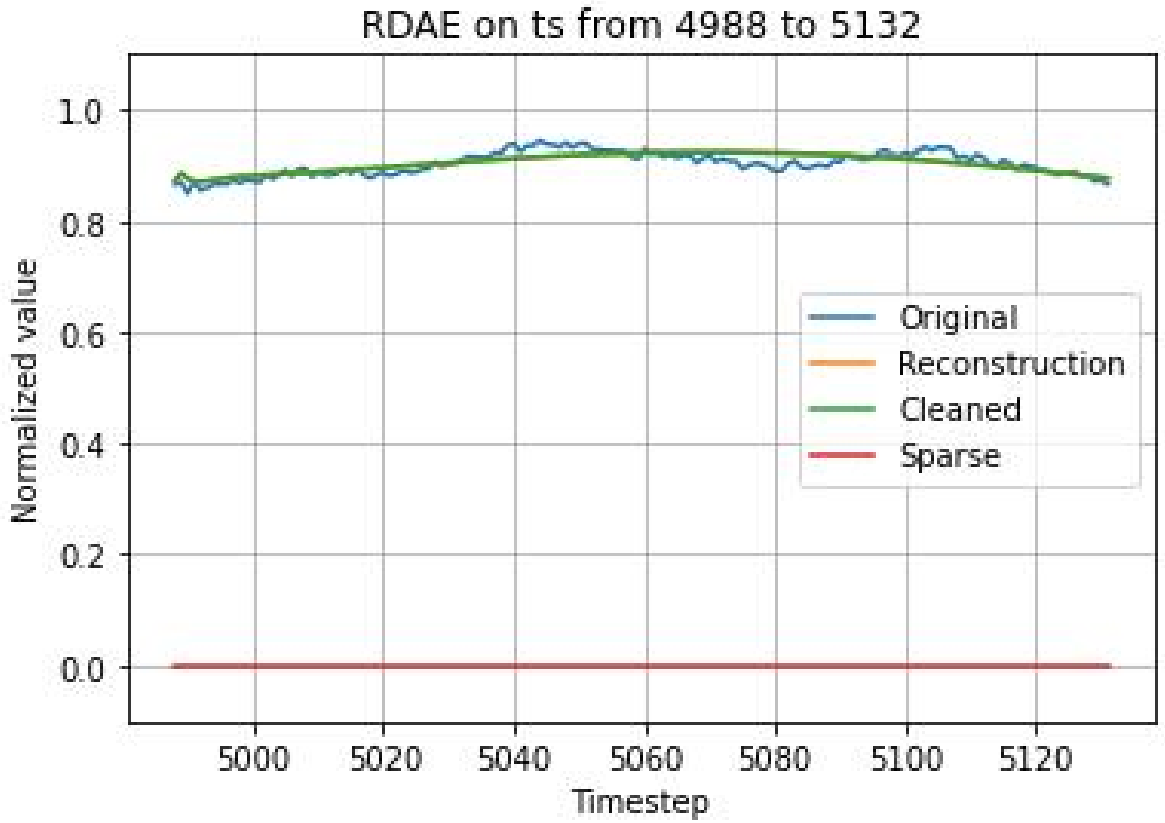



Figure: Example of a non anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

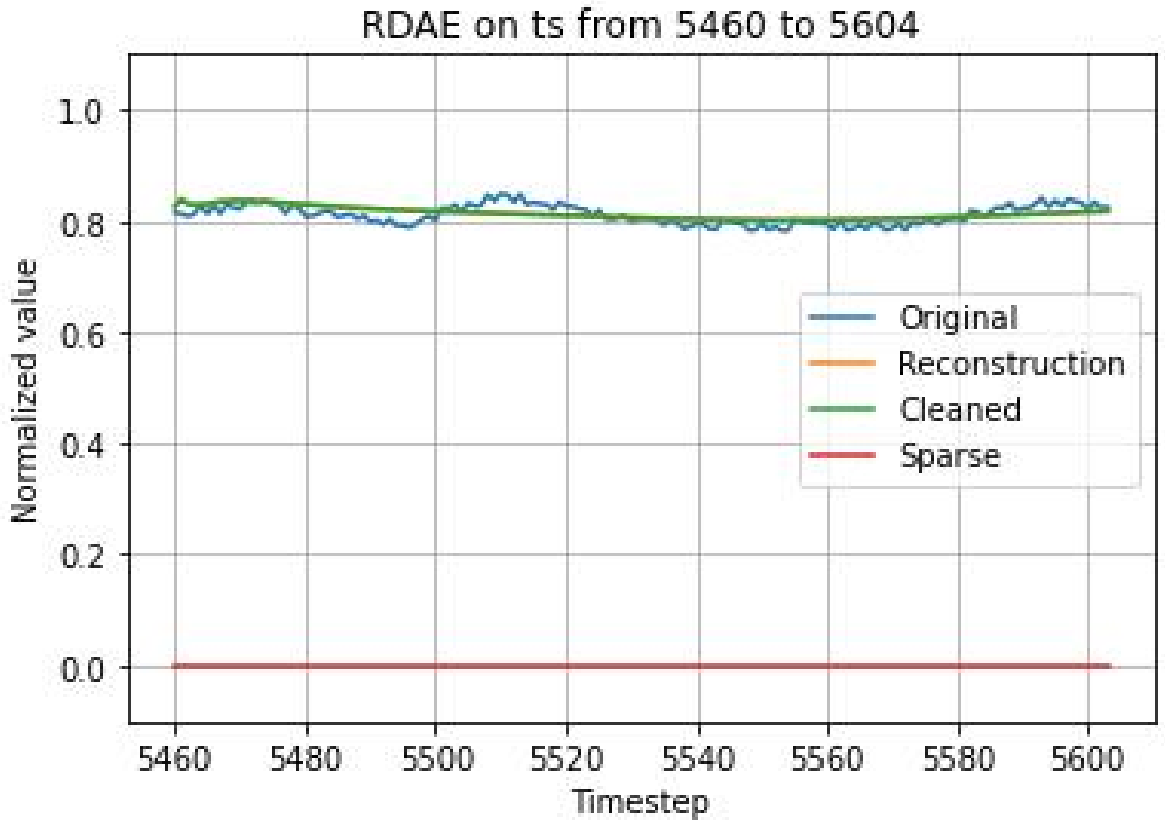



Figure: Example of a non anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

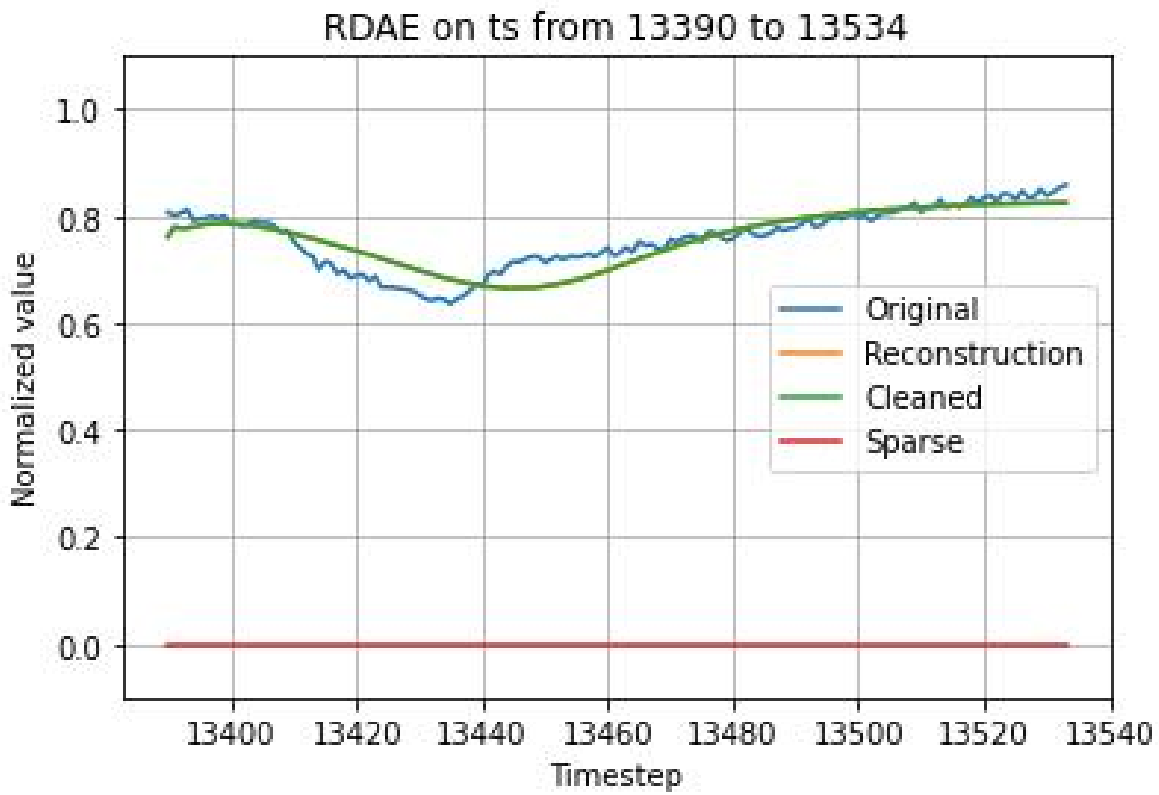



Figure: Example of a non anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

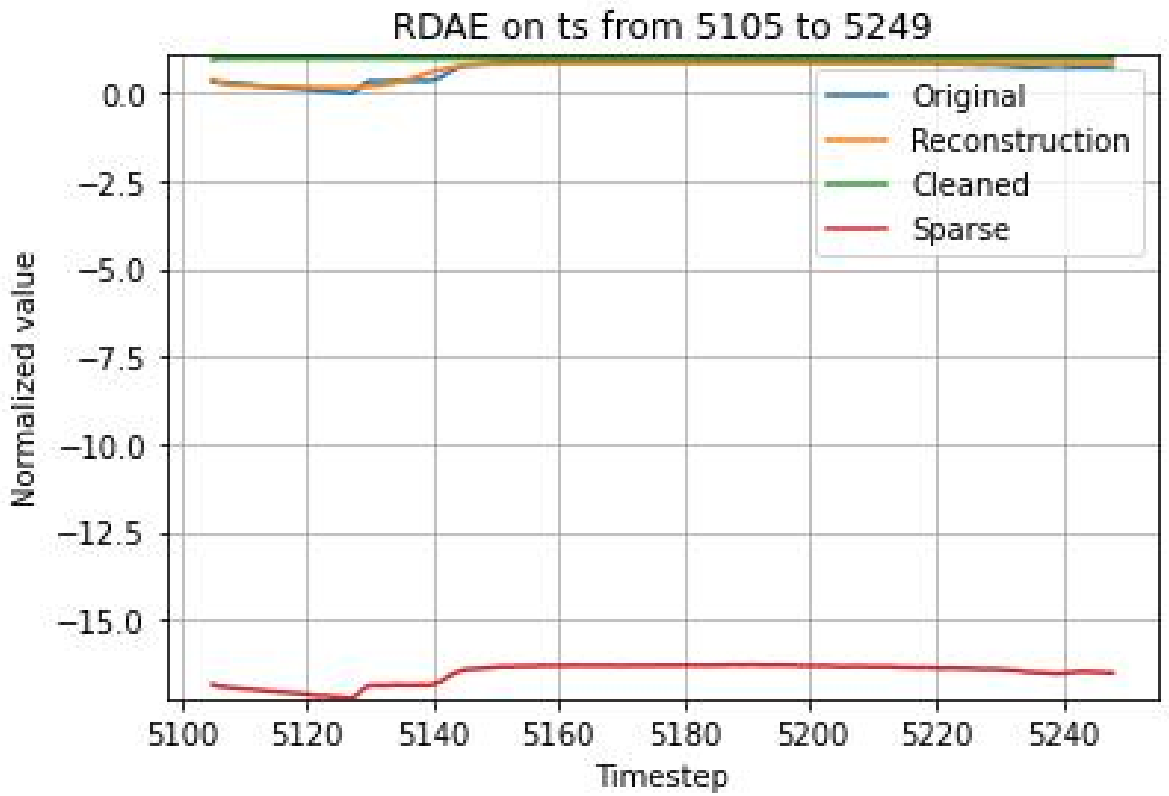



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

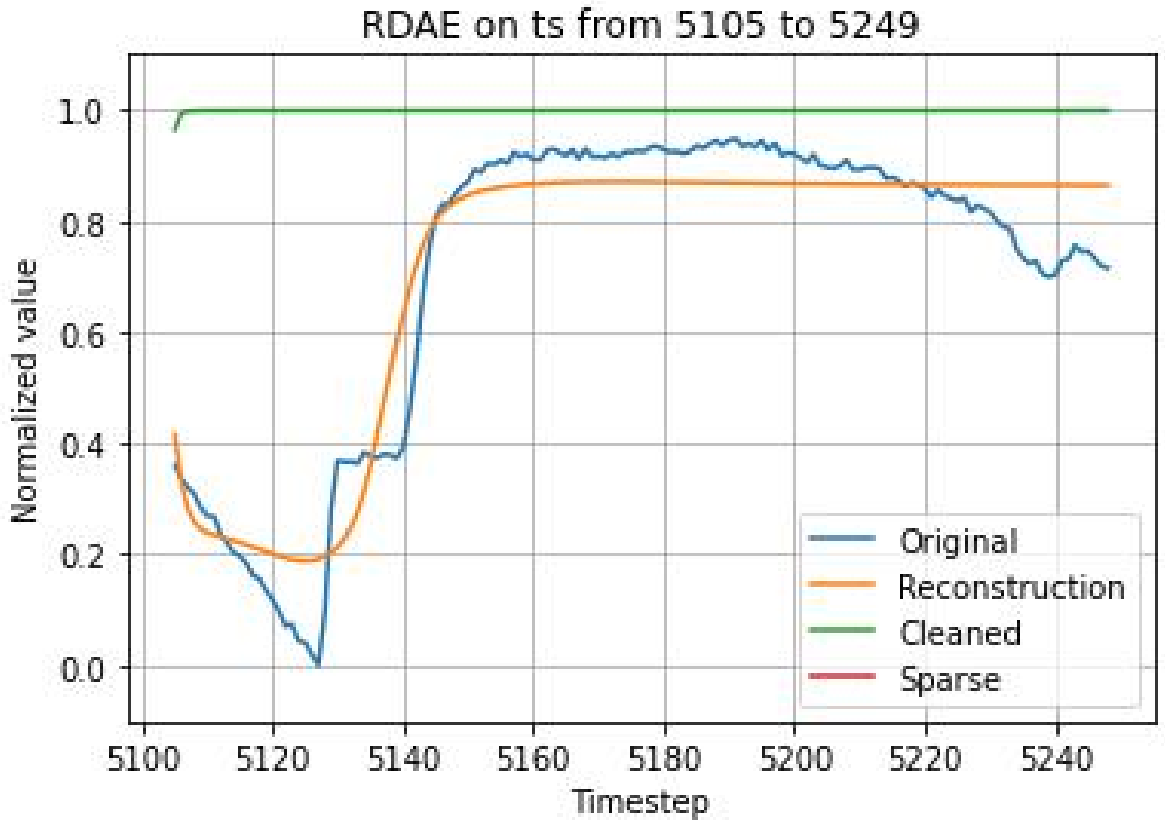



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

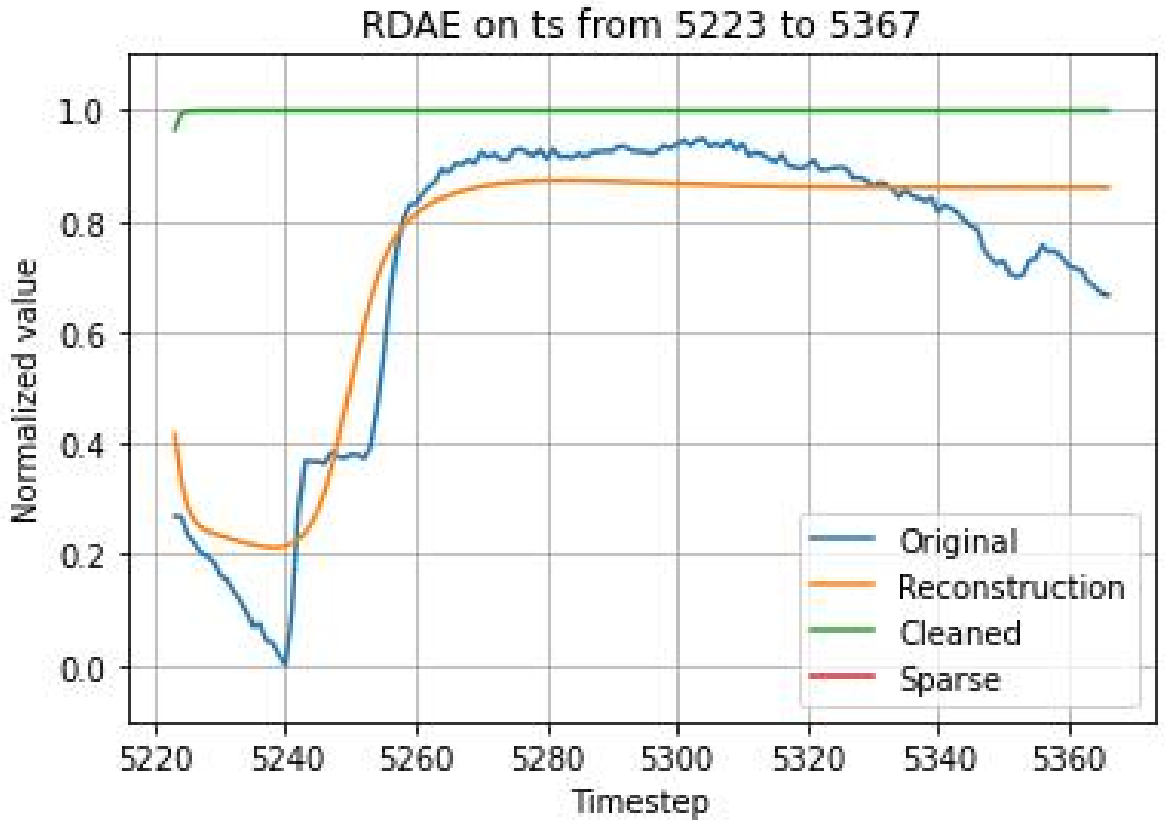



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

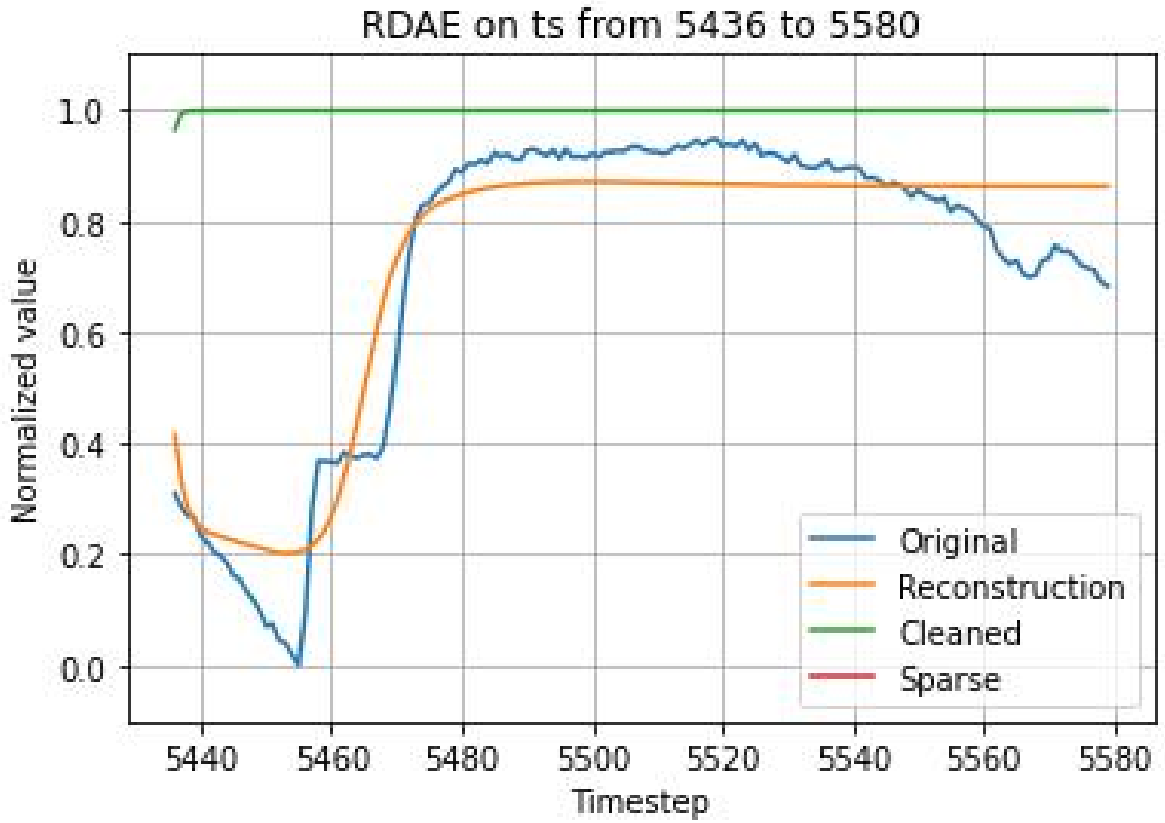



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# LSTM RDAE

- All of the anomalies found reach the 0 value (min temperature of all time series).
- Also in this case different anomalies found DO NOT overlap. So each of the failures is only recognized once. In this case this happens at the beginning of the subsequence
- The reconstruction here is far worse than in the dense case.
- Performance could be improved using more parameters in the LSTM case. Note that computation time is much higher ($\sim 4$ minutes for dense, $\sim 20$ minutes for LSTM, with the help of a RTX3070 laptop).

# GRU RDAE

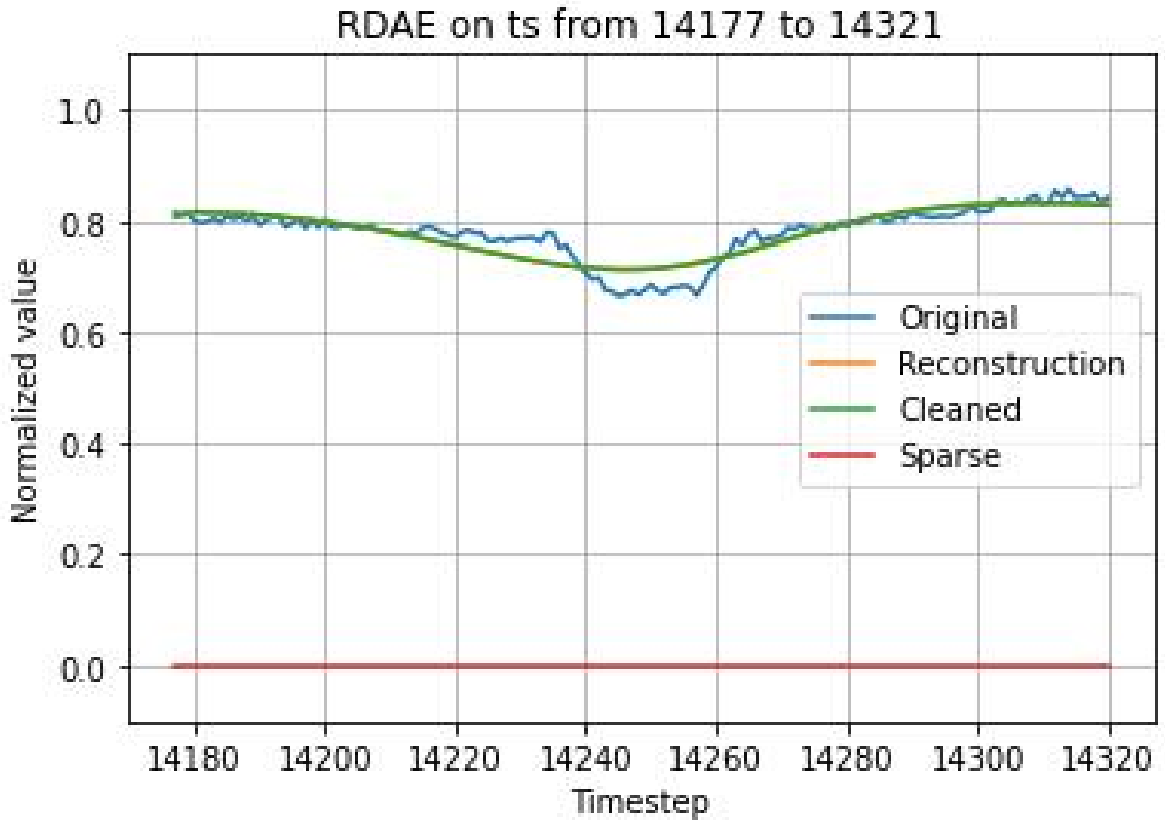



Figure: Example of a non anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

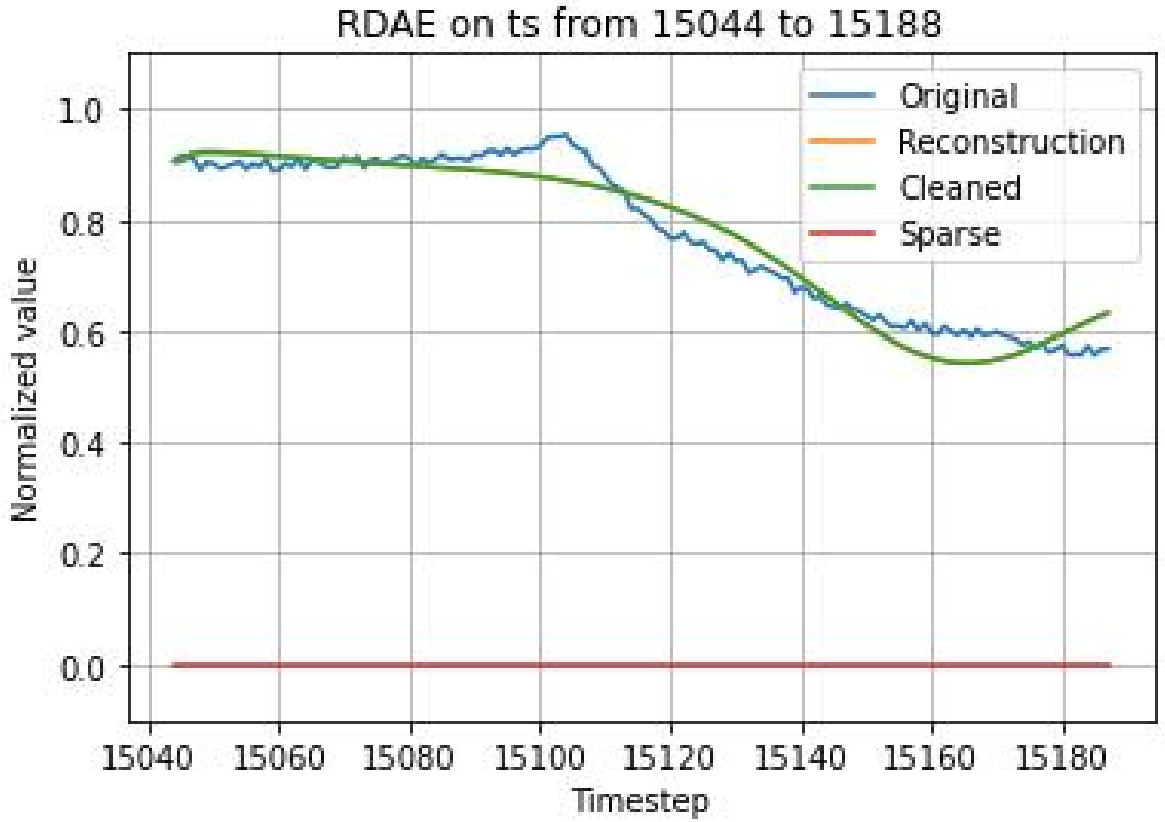



Figure: Example of a non anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

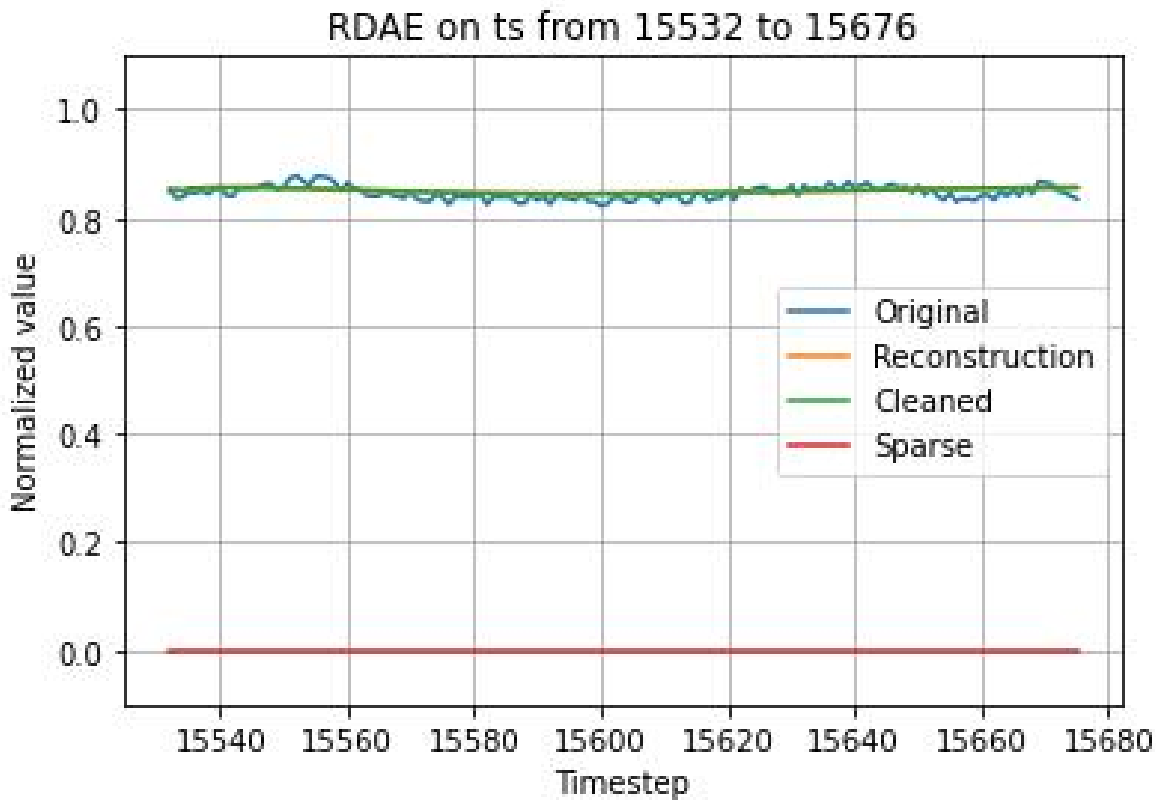



Figure: Example of a non anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

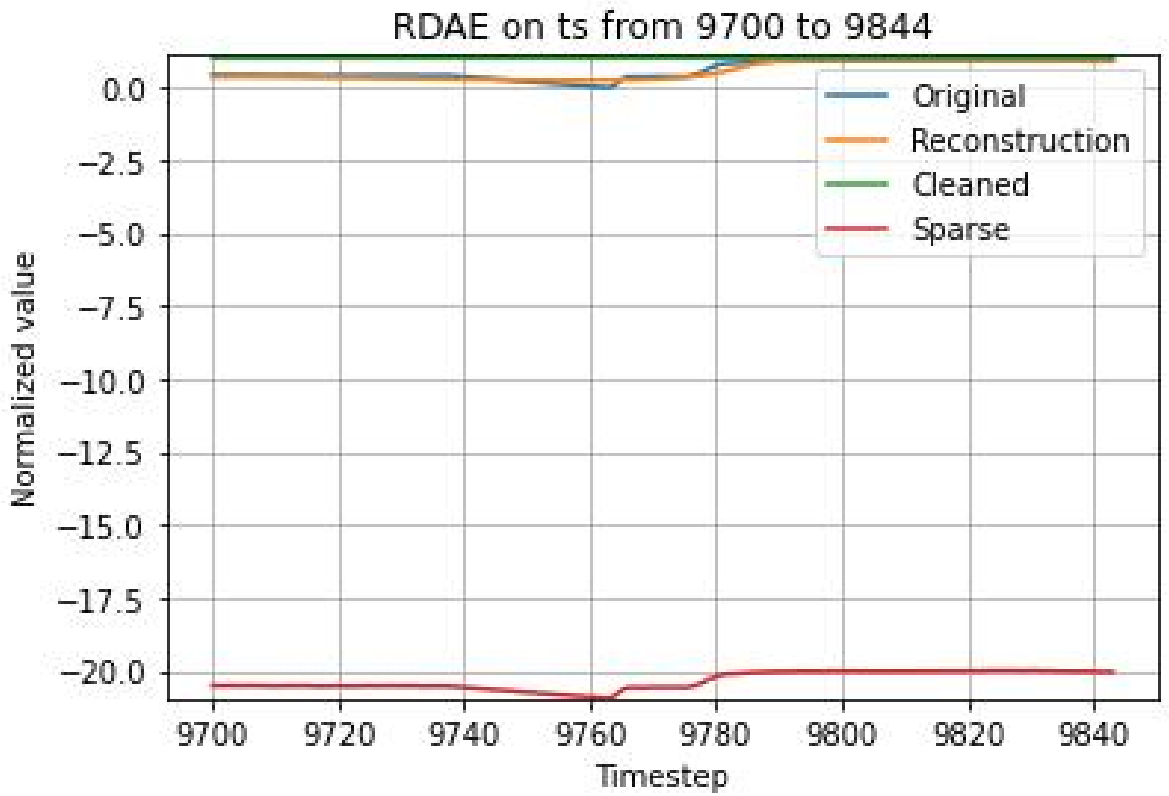



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

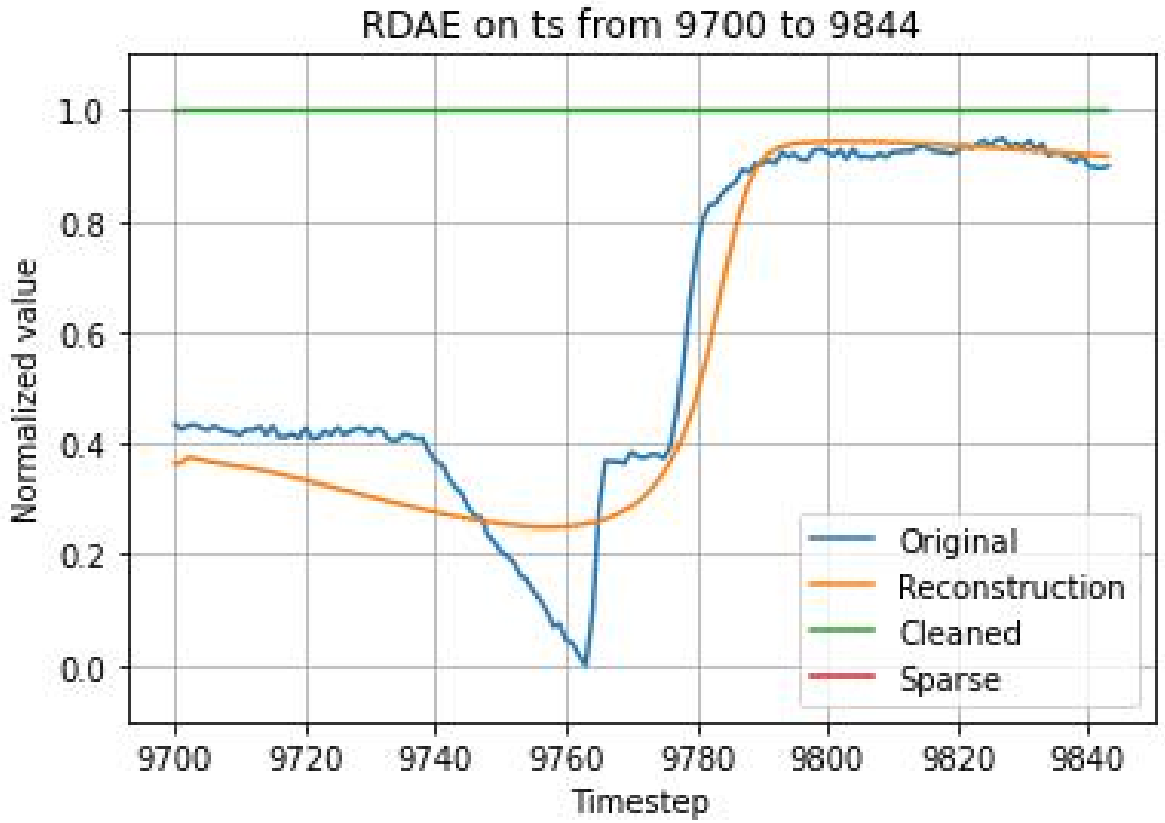



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

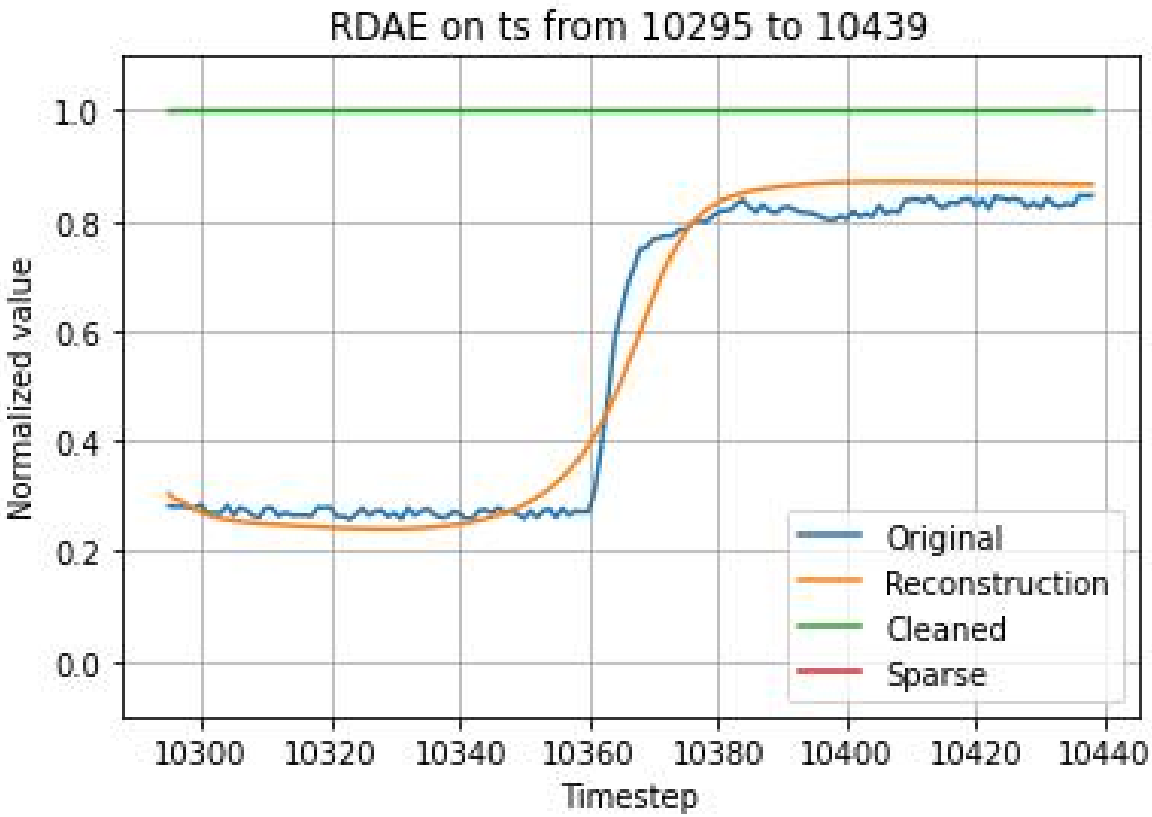



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

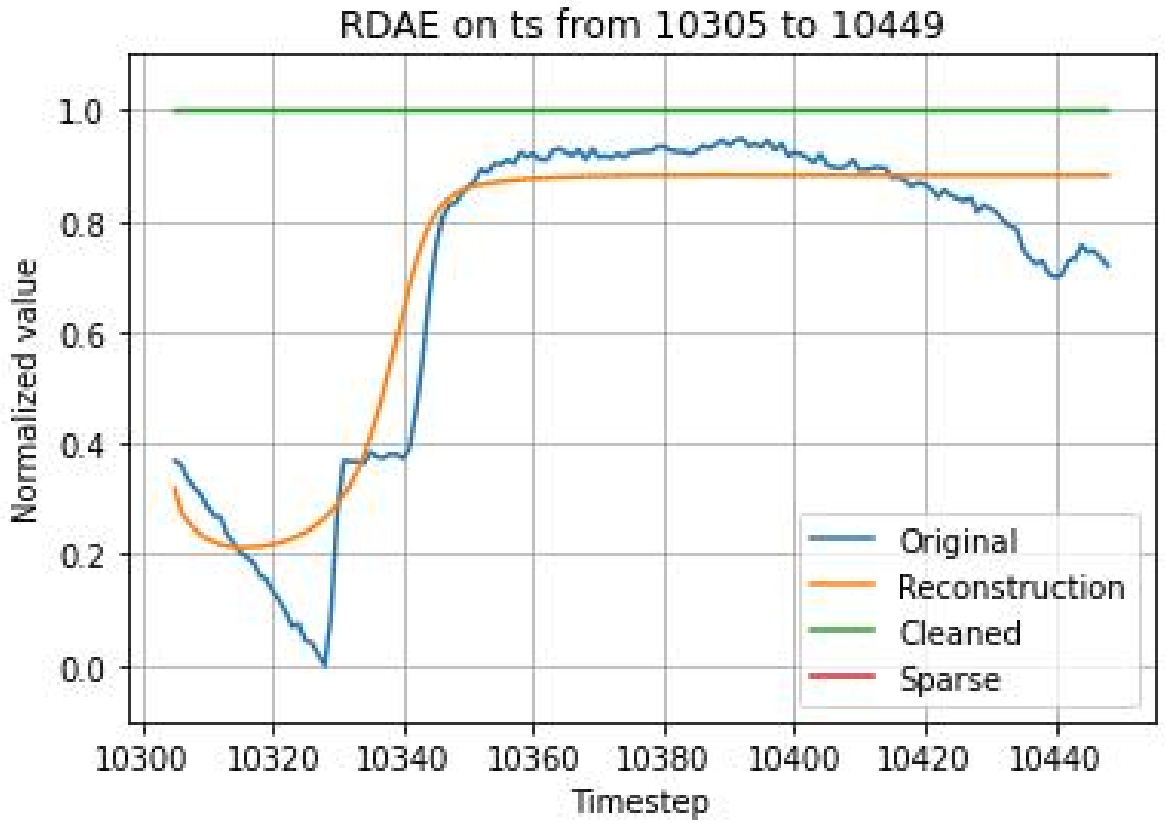



Figure: Example of a anomaly subsequence for $\lambda = 3.3$

# GRU RDAE

- In this case, not all of the anomalies found reach the 0 value.
- Also in this case different anomalies found DO NOT overlap. So each of the failures is only recognized once. With GRU this happens in different parts of the subsequence.
- Also here the reconstruction is far worse than in the dense case.
- Again we could increase parameters for better performance. Computation here required $\sim$ 12 minutes with GPU.

# Final comments

- The RDAE is a powerful tool for denoising and anomaly detection.
- 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 where 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.

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.