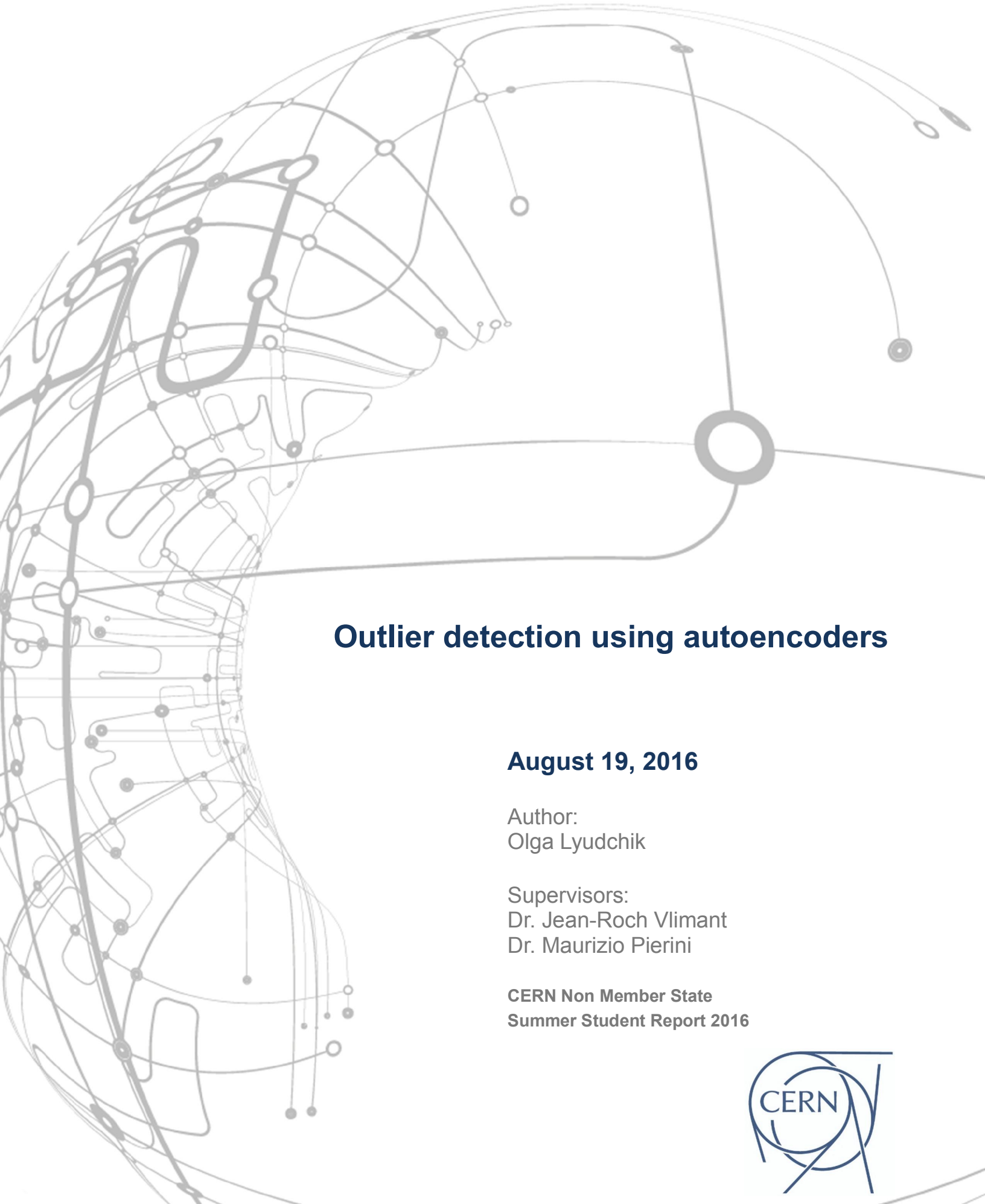


Outlier detection using autoencoders

August 19, 2016

Author:
Olga Lyudchik

Supervisors:
Dr. Jean-Roch Vlimant
Dr. Maurizio Pierini

CERN Non Member State
Summer Student Report 2016

Abstract

Outlier detection is a crucial part of any data analysis applications. The goal of outlier detection is to separate a core of regular observations from some polluting ones, called “outliers”. We propose an outlier detection method using deep autoencoder. In our research the invented method was applied to detect outlier points in the MNIST dataset of handwriting digits. The experimental results show that the proposed method has a potential to be used for anomaly detection.

Table of Contents

Introduction

The majority of data analysis tasks require a large number of variables to be recorded. One of the first steps towards obtaining a coherent analysis is to detect the outlaying observations. An anomaly or outlier is a data point, which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism. Analyzing and detecting anomalies is important because it reveals useful information about the characteristics of the data generation process. Outlier detection is applied in network intrusion detection, medical diagnosis, sensor network fault detection, credit card fraud detection and numerous other fields [1].

Existing data mining approaches to outlier detection usually rely on notions of distance and density [2]. Distance-based approaches define an instance to be an outlier in case that it is sufficiently far from the majority of other instances in the dataset. Density-based approaches consider an instance to be an outlier if its density is sufficiently small relative to the average density of its neighboring instances. Techniques based on these approaches typically explore outliers in their original data spaces (or sometimes in projected subspaces) and have demonstrated to work well on linearly separable distributions with stable densities. However, they tend to underperform with nonlinear structures with variation in densities.

Unlike the group of techniques mentioned above spectral anomaly detection techniques try to find the lower dimensional embeddings of the original data where anomalies and normal data are expected to be separated from each other. After finding those lower dimensional embeddings, they are brought back to the original data space which is called the reconstruction of the original data. By reconstructing the data with the low dimension representations, we expect to obtain the true nature of the data, without uninteresting features and noise. Reconstruction error of a data point, which is the error between the original data point and its low dimensional reconstruction, is used as an anomaly score to detect outliers. One of the most common methods for outlier detection of this group is Principal components analysis (PCA). Although it is not their usual purpose, autoencoders can also be used for dimension reduction and anomaly detection[3]. By reducing the number of nodes in the hidden layer, it is expected that the hidden units will extract features that well represent the data. In this report we propose an anomaly detection method using deep autoencoders. A deep autoencoder is composed of two deep-belief networks and allows to apply dimension reduction in a hierarchical manner, obtaining more abstract features in higher hidden layers leading to a better reconstruction of the data.

1. Background: Deep Autoencoder

A deep autoencoder is an artificial neural network, composed of two deep-belief networks that typically have four or five shallow layers representing the encoding half of the net, and second set of four or five layers that make up the decoding half.

The layers are restricted Boltzmann machines, the building blocks of deep-belief networks, with several peculiarities. A simplified schema of a deep autoencoder's structure is shown in Figure 1.

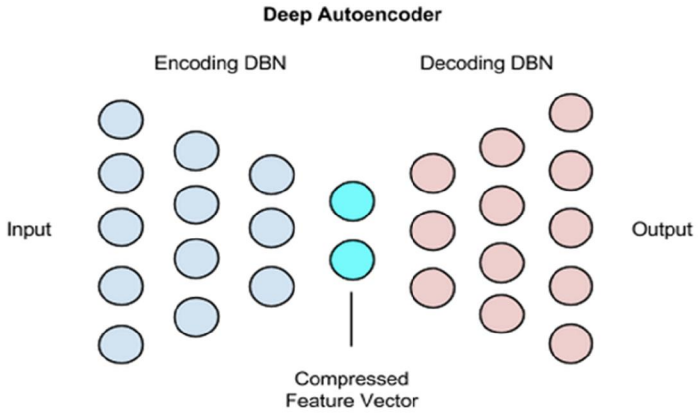


Figure 1. Schema of a deep autoencoder

2. Experimental Results

In this chapter, we will examine the general characteristics of the dataset used in our work, describe the topologies of networks being used in experimental part and provide results of the research work

2.1. Dataset

For anomaly detection we used MNIST dataset provided by Keras (a highly modular neural networks library, written in Python) [4]. MNIST is a simple computer vision dataset. It consists contains 60,000 digits ranging from 0 to 9 for training the digit recognition system, and another 10,000 digits as test data. Each digit is normalized and centeredin a gray-level image with size of 28x28. Some examples are shown in Figure 2.

Figure 2. Examples of MNIST data set

Since each image has 28 by 28 pixels, we get a 28x28 array. We can flatten each array into a 28∗28=784 dimensional vector. Each component of the vector is a value between zero and one describing the intensity of the pixel. Thus, we generally think of MNIST as being a collection of 784-dimensional vectors.

2.2. Topologies of networks being used

For our research, we created 39 different models of deep autoencoders. The encoder and decoder parts are both symmetrical deep-belief networks with the latent dimension varied from 2 to 128 neurons and the number of hidden dense layers varied from 1 to 5. Each next layer was trained by stacking the previous layer output. For all the created models for the last output layer with 784 neurons we used sigmoid as an activation function. Relu activation function was used for all the rest output layers, including a layer with latent dimension. To build our autoencoders we used 'mean square error' loss function and 'adadelta' optimizer.

2.3. Training the models

As the goal of the research was to find a way to use autoencoders for outlier detection, we first should define an outlier and normal data. We set '7' as an outlier digit by removing it from the training and testing datasets. Then we trained our models on normal data using earlyStopping mechanism from Keras callbacks. The size of MNIST datasets for outlier and normal data is shown in Table 1.

Table 1. Size of MNIST dataset

Normal data	*Training set*	53,735 samples
	Testing set	8,972 samples
Outlier	*Training set*	6265 samples
	Testing set	1028 samples

2.4. Data visualization

For the models with 2 latent dimensions we are able to visualize our data. The result of visualization for the autoencoder with 5 hidden layers is shown below in Figure 3. Each of these coloured clusters is a type of digit. Close clusters are digits that are structurally similar (i.e. digits that share information in the latent space).

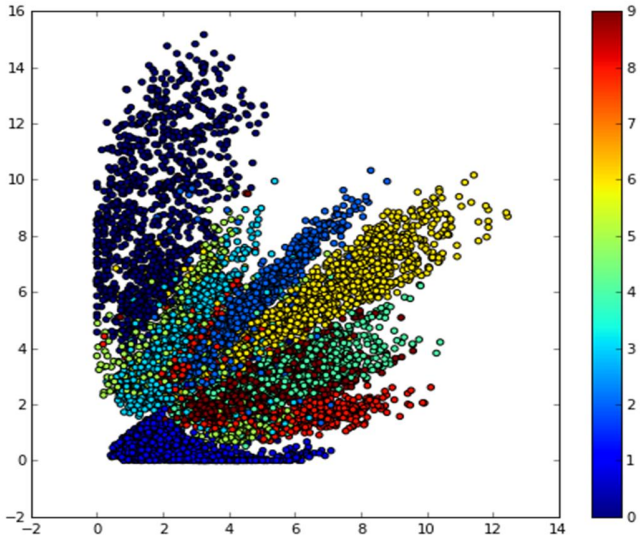


Figure 3. Plot of the digit classes in the 2-dim space for the model with 5 hidden layers

2.4.1. Visualisation of digits' transformation along the line

For the model with 5 layers we saw that cluster of 8 digits (red one on the plot) has been separated into parts. In order to understand why it had happened we visualized some digit samples along the line crossing both parts of cluster with 8 digits. In Figure 4, there are results of digits transformation, which occurs along the given 'pink' line in the arrow direction the 8 digit.

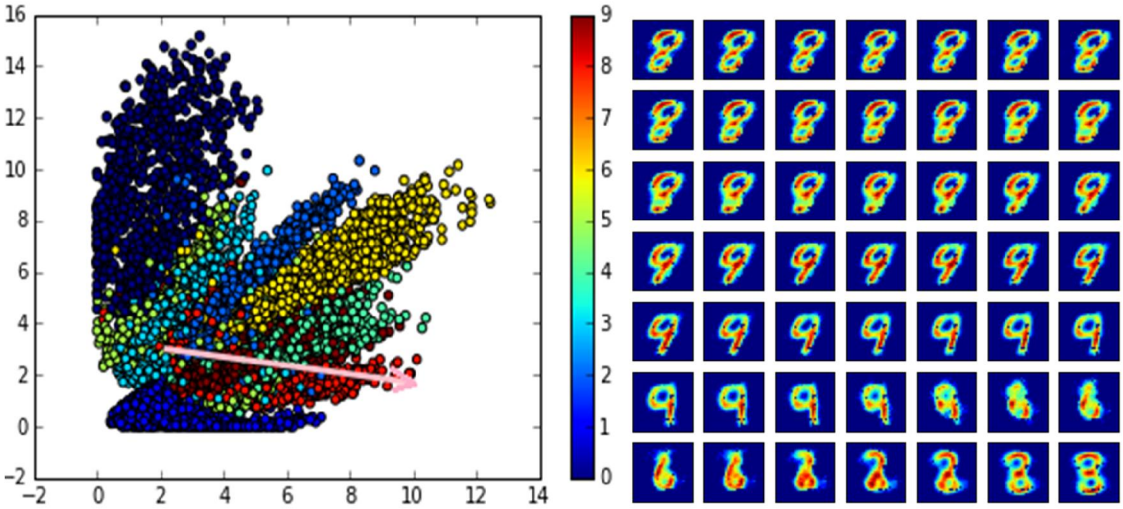


Figure 4. Transformation of digits along the line

The 8 digit transforms gradually to 9 and then through some states turns back to the figure resembling 8 digit. These results can be explained by the similarities in the structure of 8 and 9 digits.

2.4.2. Visualisation of digits' transformation around a point

Another way to have a look on how different digits are connected to each other is to choose one data point and plot some digit samples around it. This can be one of the ways to understand why some digits are not classified correctly. Look at the example. The red arrow in Figure 6 below pointed to the 8 digit, which reconstruction is shown in Figure 5. This digit was classified to the wrong cluster of zeros.

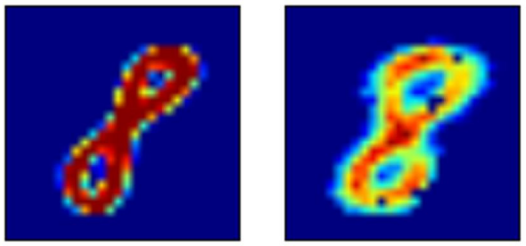


Figure 5. 8 digit and its reconstruction

The results of digit transformation around 8 digit with the radius r = 0.2 are shown in Figure 6.

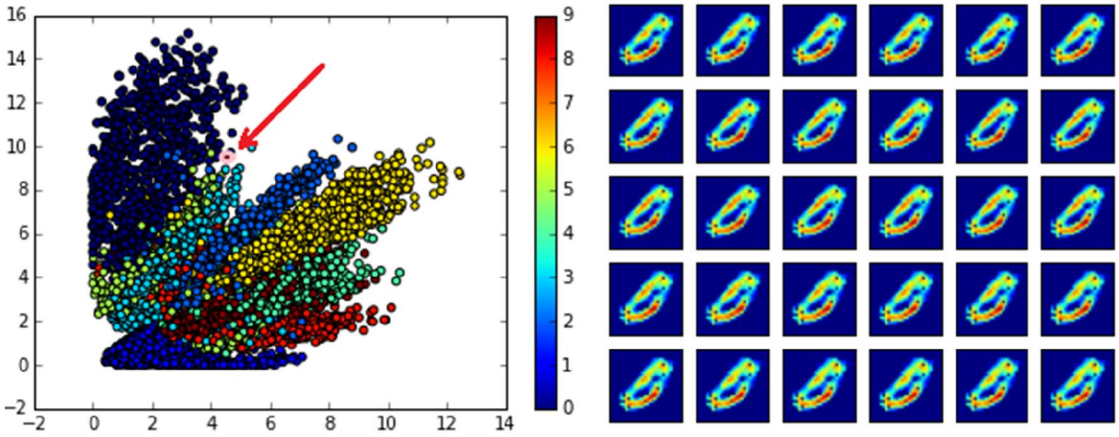
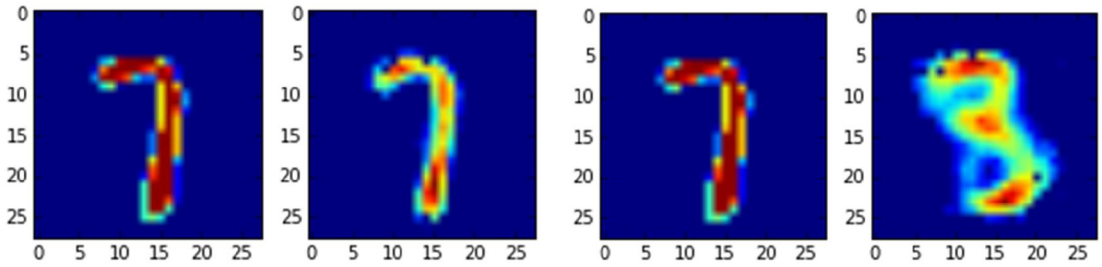


Figure 6. Transformation of digits around the 8 digit

The possible explanation of 8 being classified to the wrong group is that 0 digits in Figure 6 on the right and the 8 digit have a very similar structure and the model found them more similar to each other than the 8 with digits from original cluster of 8.

2.5. Reconstruction of outlier

The next consistent step was to test our models on outlier reconstruction. The results of data samples and its reconstructions for anomaly digit are shown on figures below. Figure 7 (a) reflects the highest reconstruction probability which corresponds to the model with 3 hidden layers and 64 latent dimensions. The mean difference between the input image and the output one for this network is 0.344363. The lowest reconstruction probability with the mean difference between the input image and the output one of 0.944811 corresponds to the model with 2 hidden layers and 2 latent dimensions and is presented in Figure 7 (b).



a) Model with 3 hidden layers and 64 latent dimentions

b) Model with 2 hidden layers and 2 latent dimentions

Figure 7. Data samples and its reconstructions for anomaly digit

It is clearly seen in the Figure 4 (a) that even though 7 was in the anomaly data and not given to the training data the network seems to represent 7 with high reconstruction probability. The possible explanation of that phenomenon can be that because the structure of 7 being a single vertical stroke with a horizontal stroke on the top is so simple, the autoencoder learned that structure from the other parts of the data. For example, a vertical stroke is included in 1, in the centre in 4 and 9 if

written in a rigid manner without much curvature. The horizontal stroke on the top is used in almost every digit with the exception of 1 and 4. These features might have given data for the deep autoencoder to learn the components of structures.

2.6 Performance evaluation

To evaluate the ability of our models to detect outliers we made histograms of mean difference between the prediction of the model and its original input for normal and outlier data. What we expected to get was the visible shift of outlier data in relation to normal one. However, most of our models did not seem to perform in such a way. The most acceptable result was given by the deep autoencoder with 3 hidden layers and 6 latent dimensions (Figure 8).

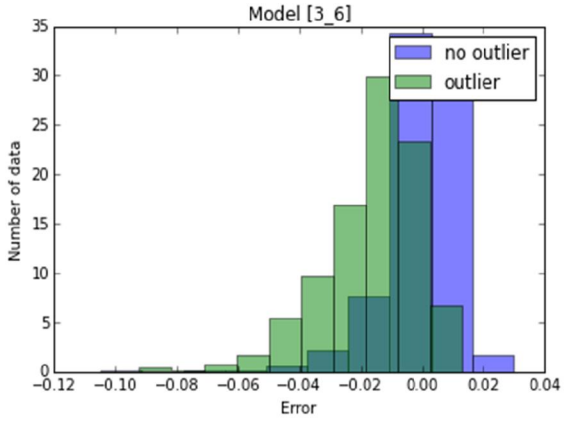
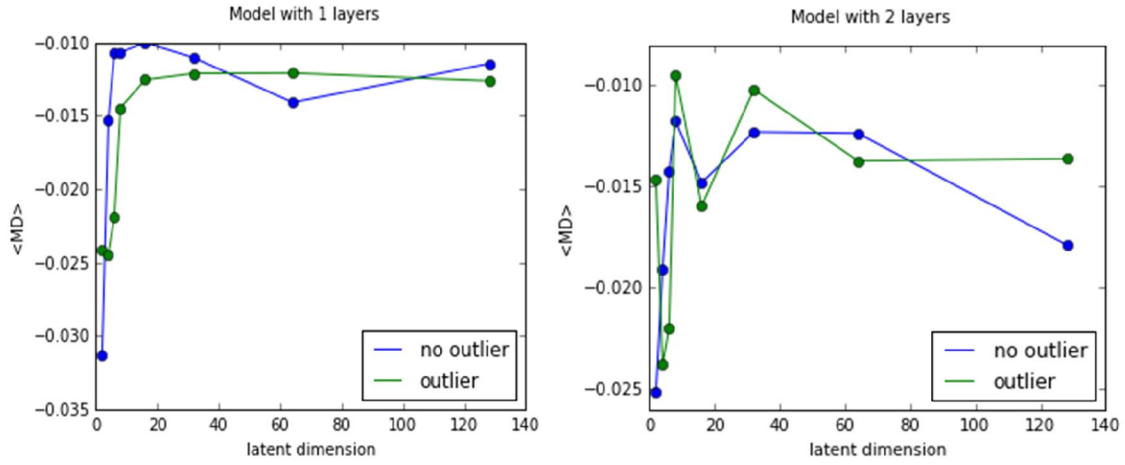


Figure 8. Error distributions for normal and outlier data produced by the model with 3 hidden layers and 6 latent dimensions

To see how the mean difference depends on latent dimension and number of layers we made two groups of plots. The model with best performance is circled in red. The first group shows the dependence of average mean difference on the latent dimension for models with a fixed number of layers (Figure 9).



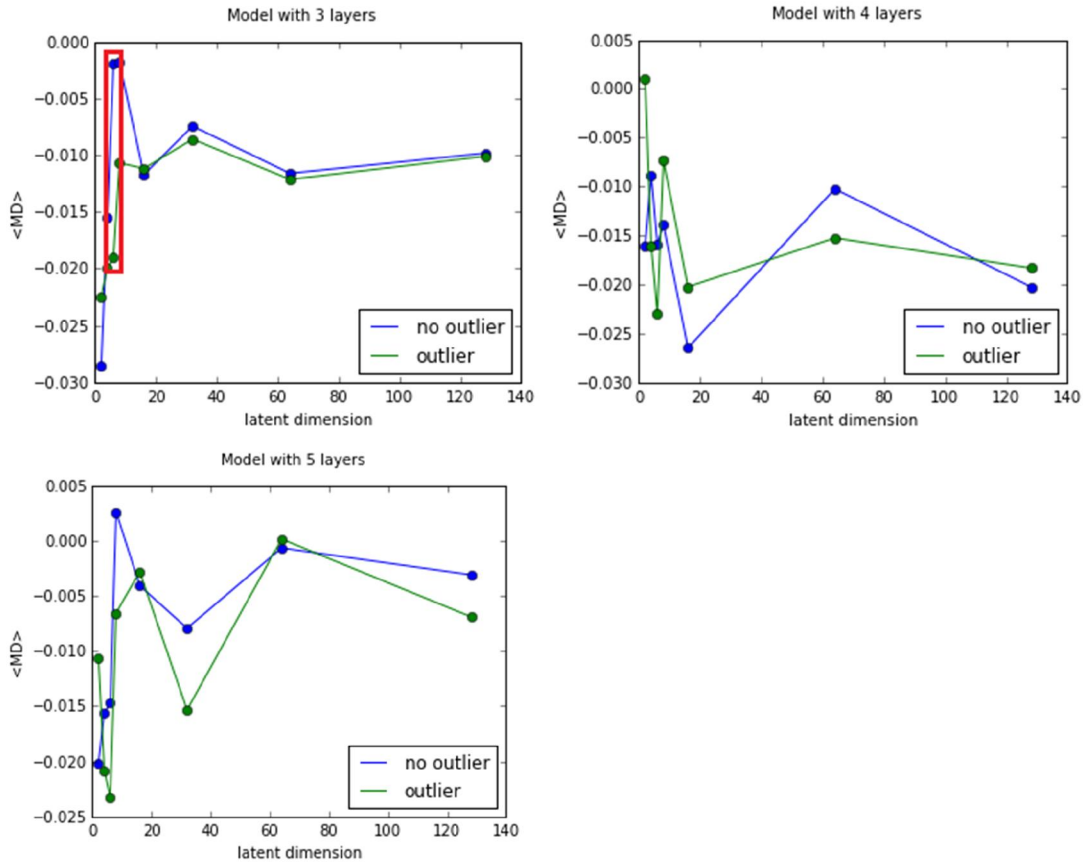
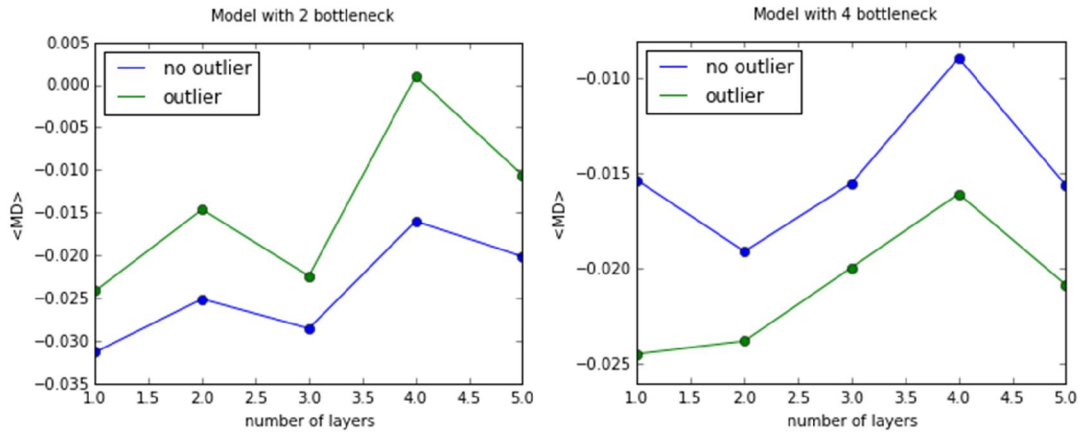


Figure 9. Dependence of average MD on latent dimensions for a fixed number of layers

The second group of plots represents the dependence of average mean difference on the number of hidden layers for models with a fixed latent dimensions (Figure 10).



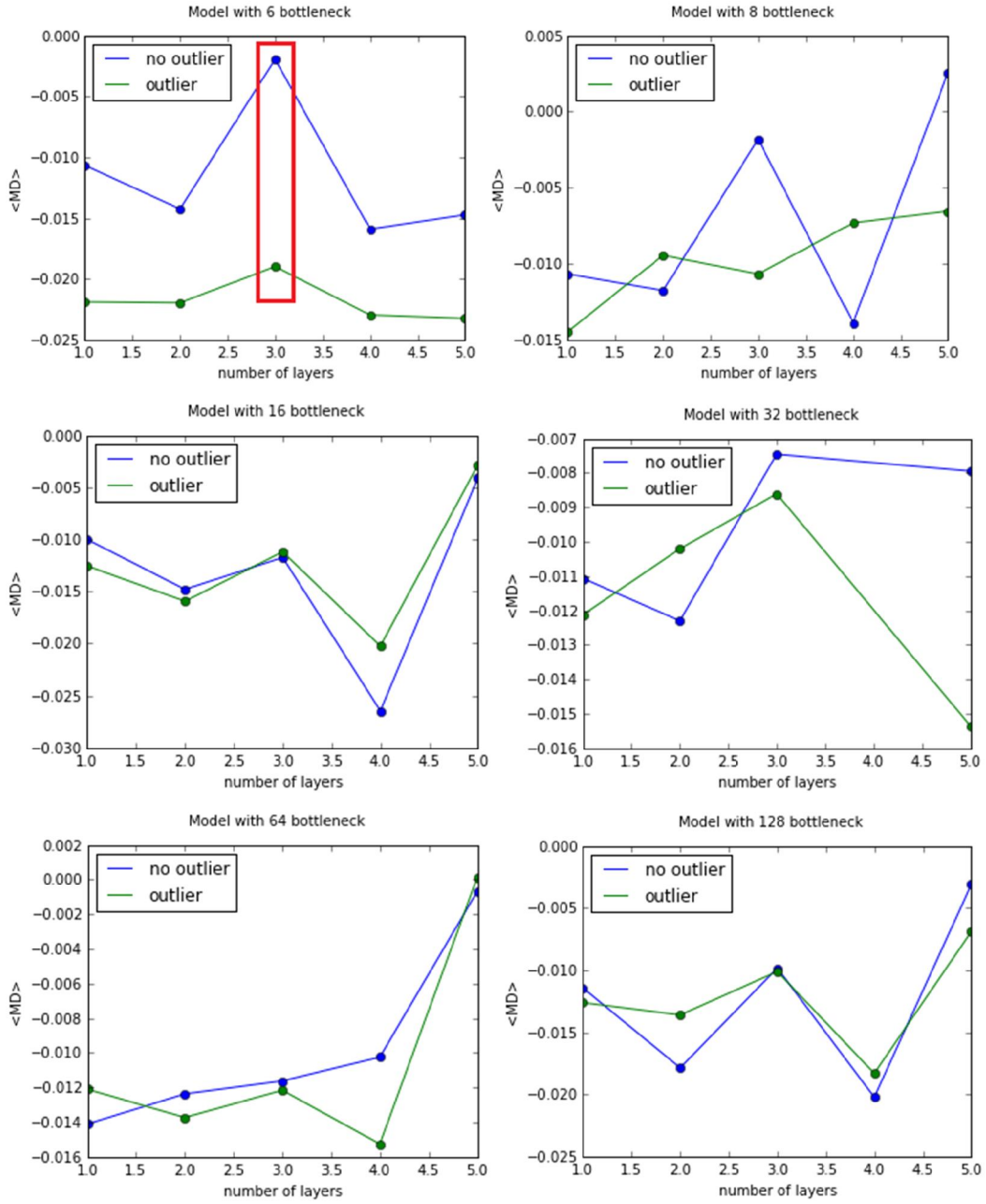


Figure 10. Dependence of average MD on the number of hidden layers for a fixed latent dimension

Conclusion

Although the results reveal some fluctuations, the introduced method has a potential to be used for outlier detection. Several models have shown promising outputs. Some of the following steps can be done in order to obtain more accurate results. To define the ideal network topology of deep autoencoder we can retrain existing models several times, test some new models with different amount of hidden layers and latent dimensions. We can try other loss functions and use another activation functions for output layers to achieve more reliable results.

References

1. Karanjit Singh and Dr.Shuchita Upadhyaya "Outlier Detection: Applications and Techniques" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 3, January, 2012

2. Hans-Peter Kriegel, Peer Kröger, Arthur Zimek "Outlier Detection Techniques" Tutorial at 10th SIAM International Conference on Data Mining (SDM 2010), Columbus, OH, 2010

3. UFLDL Tutorial. http://ufldl.stanford.edu/wiki/ index.php/Autoencoders and Sparsity.

4. Keras datasets https://keras.io/datasets/