18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

# A Novel Approach for Trajectory Feature Representation and Anomalous Trajectory Detection

**Wenhui Feng, Chongzhao Han**
MOE KLINNS Lab, Institute of Integrated Automation
School of Electronics and Information Engineering
Xi'an Jiaotong University
Xi'an, China 710049
Email: wenhfeng@stu.xjtu.edu.cn, czhan@mail.xjtu.edu.cn

***Abstract – Trajectories obtained from low level tracking algorithm provide an opportunity for us to analyze meaningful behaviors and monitor adverse or malicious events. How to abstract meaningful features from the raw data of trajectories is a challenge due to the high dimensionality and noise. In this paper, a novel approach, stacked denoising autoencoder(SDA) is applied to address this problem. This method can reduce the dimensionality of the trajectories significantly, so that they can be handled easily. More importantly, the denoising process of the SDA can capture the structure of the raw data, so the features they producing generalize well for detecting anomalous trajectories. The results of the numerical experiments prove the validity of the proposed approach.***

**Keywords:** anomalous trajectory detection, feature representation, stacked denoising autoencoder

## 1 Introduction

Anomaly detection has been studied by a variety of methods and in the context of a large member of application domains. Several extensive surveys on anomaly detection are available [1, 2]. As far as safety surveillance systems are concerned, detection and tracking moving object, then analyzing and predicting its behavior to detect anomalous target trajectory is the conventional roadmap for many works [3-5].

The raw data of trajectories are a list of vectors with elements representing the spatial position and velocity of the objects. So they are with very high dimensionality and often with noise. Finding a proper representation for the data is always of great importance. A variety of feature extraction and dimensionality reduction techniques have been applied to address the problem of trajectory representation. Johnson and Hogg[5], Stauffer and Grimson[6] presented vector quantization to address this problem, while wavelet coefficients were applied in [7, 8]. And there were some other authors employed component analysis and DFT to deal with the feature representation problems.

When tracking a moving object, the conventional tracking algorithm (such as Kalman filter, particle filter and IMM) will give an estimated variance for the estimator. The authors believe that fusing this information in the representation algorithm will improve the accuracy of the detection. Yet as far as the authors know, few of the available methods employed for trajectories representation have concerned the uncertainty of the raw data thoroughly or using the estimated variance effectively.

The aim of the trajectory representation is to find a proper feature extraction algorithm which can reduce the dimensionality of the raw data and is robust to the noise of the raw data. In this paper, we present a novel method, stacked denoising autoencoder(SDA), to abstract features from the raw data. SDA is an important algorithm recently invented in deep learning. Deep learning, also known as representation learning, is one of the hot topics in machine learning and intelligent computing communities. The main idea of denoising autoencoder is to learn representations robust to corruption of the input pattern by adding noise to the raw data artificially and making the decoder get an output as close as possible to the uncorrupted input.

The remainder of the paper is organized as follow: Section 2 briefly reviews the denoising autoencoder algorithm. Section 3 discusses the possibility of applying this technique to trajectories representation and gives some preliminary examples. Then, some experimental results proving the validity of the proposed approach is given in Section 4. Finally, Section5 concludes the study.

## 2 Denoising Autoencoder

The main idea at the basis of denoising Autoencoder was first proposed by Vincent et al [9]. A good representation should capture stable structures of the raw data regardless they are contaminated. For high dimensional input (such as the trajectory), such structures are abstracted from a combination of many elements of the input vector, and should remain unchanged (or changed only slightly) when the input data are noisy.

### 2.1 The Basic Autoencoder

We begin with the basic autoencoder. An autoencoder is a typical unsupervised learning algorithm. In its simplest form, an autoencoder has two parts, an encoder and a decoder. When the dimension of the encoder's output is smaller than the input dimension, it can be seen as a technique of

dimensional reduction. Yet in its modern instantiation, the output dimension can also be higher than the input dimensionality. In this case, it is seen as an over-complete representation.

We denote the training set as $X = \{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$, where $x \in R^d$, $d$ is the dimensionality of the input vector, and $m$ is the number of the training vectors.

The encoder is a mapping $f(\cdot)$ which transforms an input vector $x$ into a hidden representation $z$. Its typical form is an affine mapping followed by a nonlinearity $s(\cdot)$:

$$z = f(x) = s(Wx + b) . \quad (1)$$

The output vector takes the form of $z \in R^p$, $p$ is the dimensionality of the hidden representation; the weight matrix $W$ is an $R^{p \times d}$ matrix, and $b$ is an offset vector of dimensionality $p$. The function $s(\cdot)$, also known as activity function, takes the form of the family of sigmoid function. A typical choice is logistic function

$$s(x) = \frac{1}{1 + \exp(-x)} . \quad (2)$$

Clearly, the range of this function is $(0,1)$.

The decoder function $g(\cdot)$ maps the hidden representation $z$ back to a reconstructed vector $y$ in input space

$$y = g(z) = h(Dz + c) . \quad (3)$$

The reconstructed vector $y$ has the same dimensionality as the input vector $x$. So the weight matrix of the decoder $D$ is a $d \times p$ matrix. When $D$ is constrained by $D = W^T$, the autoencoder is said to have tied weights. And $c \in R^d$ is the offset vector of the decoder. The activity function of the decoder typically is either the identity (yielding linear reconstruction) or a sigmoid. It can be seen that when logistic function is taken as an activity function, rang of the reconstructed vector is constrained as $y \in [0,1]^d$. The aim of the autoencoder is to make the reconstructed vector close to the input vector, so the input vector should be also normalized as $x \in [0,1]^d$. When the autoencoder takes the fashion of tied weights, the parameters of this model are $\theta = \{W, b, c\}$.

A variety of loss functions and algorithms have been applied to train the autoencoder. One of the primary one is as follow. The parameters are optimized to minimize the average reconstruction error:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} J(\theta) \\ &= \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}) \\ &= \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, g(f(x^{(i)}))) \end{aligned} \quad (4)$$

where $L(\cdot)$ is the reconstruction error. A typical choice of the lost function is the squared error

$$L(x, y) = \|x - y\|_2^2 . \quad (5)$$

Another choice is cross-entropy loss when $h(\cdot)$ is logistic function (in this case the input vectors are also normalized into the range of $[0,1]$ ):

$$L(x, y) = -\sum_{k=1}^{d} (x_k \log y_k + (1 - x_k) \log(1 - y_k)) \quad (6)$$

where $x_k$ denotes the k-th element of the vector $x$.

Stochastic gradient decent is a typical algorithm used to address the optimization problem. It can find a sensible local optimal value for the parameters.

## 2.2 Denoising Autoencoder

When the activity function for the encoder $s(\cdot)$ and the activity function for the decoder $h(\cdot)$ take a linear form, the autoencoder is the same as PCA. And when a nonlinear function is applied, it can extract much richer features. But just applying the reconstruction criterion alone is insufficient to guarantee that the extracted features are useful. The outputs may just simply copy the input or similarly uninteresting ones. One strategy to address this problem is to constrain the representation. For example, if we constrain that the dimensionality of the hidden variable must be less than that of the input, we get a bottleneck; and more recently, sparse representations are also a hot researching area following this strategy.

In practical applications, the input is always disturbed by noise. So it is a desirable property for the autoencoder to be robust to the noisy input. The denoising autoencoder is motivated by making the basic autoencoder robust to noise. The criterion of the denoising autoencoder is cleaning the corruption of the input, or in short denoising. Concretely, we find a stochastic mapping:

$$x \to \tilde{x} : \tilde{x} \sim q(\tilde{x} | x) \quad (7)$$

to map the input vector $x$ into a noising corrupted version $\tilde{x}$. Then the corrupted version $\tilde{x}$ instead of raw input $x$ is fed into the denoising autoencoder to compute the representation $z = f(\tilde{x}) = s(W\tilde{x} + b)$. The raw input vector $x$ is finally reconstructed by the decoder as eq. (3). The parameters of the denoising autoencoder are also trained to minimize the reconstruction error as eq. (4), that is, to make $y$ as close as possible to the uncorrupted input

vector $x$ . As the basic autoencoder, the loss function is defined as either squared error in eq. (5) or cross-entropy loss in eq. (6). The algorithm of denoising autoencoder is summered in algorithm 1.

**Algorithm 1**: Denoising Autoencoder

**Input**: training set $X$ ;
Learning rate $\eta$

**Output:** parameters of the model $\theta = \{W, b, c\}$

1: initialize parameters $\theta = \{W, b, c\}$

2: choose random minibatch $X^t = \{x^{(t_1)}, x^{(t_2)}, \cdots, x^{(t_n)}\}$

3: corrupt each vector $x^{(t_i)}$ in the minibatch according to eq. (7) to get the corrupted vector $\tilde{x}^{(t_i)}$ ;

4: compute the average reconstruction error $J(\theta)$ on the minbatch $X^t$ ;

5: update the parameters according to:

$$\theta_{new} \leftarrow \theta - \eta \frac{\partial J(\theta)}{\theta} ,$$

$$\theta \leftarrow \theta_{new}$$

6: repeat from step 2 until convergence or other terminal conditions

## 2.3 Stacked Denoising Autoencoder

Stacking denoising autoencoder(SDA) to form a deep architecture is quite the same way as stacking RBM in deep belief networks[10]. Once the mapping $f(\cdot)$ has been learned, the hidden representation $z$ is determined. Then the hidden representation $z$ can be taken as input vector for the second layer of the SDA. Following the same procedure, a deep architecture is formed from denoising autoencoder. It should be pointed out that input corruption is only used for the training of each denoising autoencoder. After the parameters of the denoising autoencoder are determined, cleaning input vectors is fed into the mapping to get the input vector for the denoising autoencoder in the next layer. A three layers' deep architecture can be shown as fig. 1

Stacking several nonlinear models to form a deep architecture is a basic idea in deep learning. It is believed

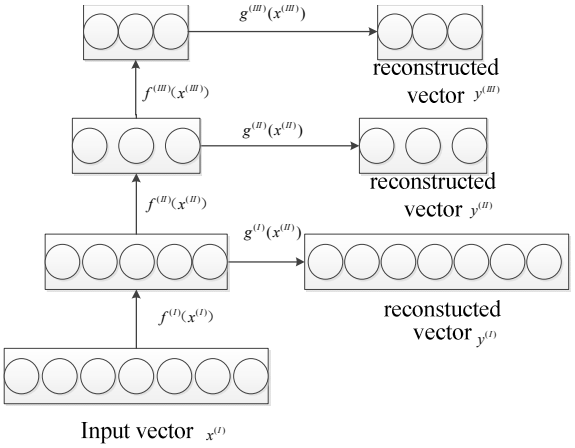


Figure 1: Three layers SDA. (The superscript Roman mumber is used to denote the number of the layer)

that the higher layer, which takes the output vectors of the lower layer as input vector, will extract more abstract features, and will achieve better generalization performance on difficult recognition tasks. This idea has been verified by many successful applications of the deep architecture in machine learning, computer vision, and pattern recognition.

# 3 SDA for Trajectory Repression

## 3.1 Possibility of SDA for Trajectory Repression

In this section, we will explore the possibility of using SDA for trajectory representation. The trajectories of the moving objects are usually obtained through low level tracking algorithm, such as Kalman filter, particle filter and IMM. These filters will estimate the position and velocity of the object in the given time instance. So the trajectory is by convention represented by a sequence of spatial position coordinate and velocity. Take 2-D space as an example, the trajectory at each time instance will be represented by 4 elements. The sensor always works at a very high rate, so tracking a moving object for just several minutes will get a trajectory represented by a vector with hundreds of elements. Dimensionality reduction is the key technique for trajectory repression.

The other problem in the raw data of the trajectory is noise. The aim of each tracking algorithm is to reduce the noise of the sensor's measurements, but no one algorithm can get rid of noise. So any algorithms must be able to work with noise. Two trajectories with difference of just little fluctuation should be regarded as the same (see Fig. 2 for an illustration).

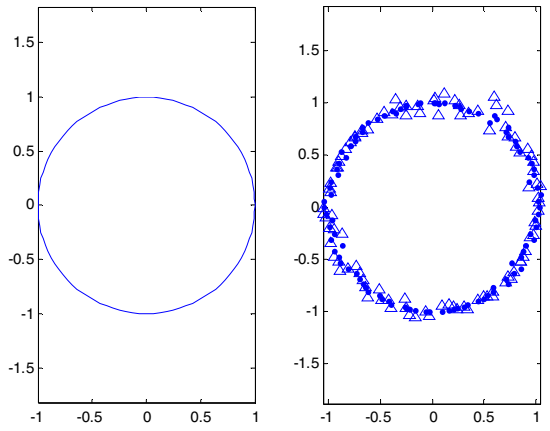

Fig.2: (Left) Illustration of a ground truth trajectory which should be a circle one; (Right) Illustration of two trajectories obtained by difference tracking algorithms; one is denoted by points and the other by triangles. And they should be regarded as the same.

Those two problems can both be addressed by SDA. For each layer of the SDA, the dimensionality of the hidden variable is set to be lower than that of the input variable. When the raw data are input into a SDA, the dimensionality is reduced by each layer of the SDA efficiently. During the training process, the raw data are corrupted by some elaborate stochastic mapping, and the reconstruction data are as close as possible to the uncorrupted input data. Thus the hidden variables are expected to be robust to noise.

## 3.2 A Preliminary Test

We now present a preliminary experiment on trajectory data representation. The experiment is motivated by Mustafa F. and Krishnamurthy V [11]. The target moves in a specific trajectory around a sensitive asset (like an embassy or a security check-point). The shape of the trajectory correlates with the intent of the target and is of great importance if it can be reliably detected. The target dynamics are summarized using its kinematic state in vector $s_t = [x_t, y_t, \dot{x}_t, \dot{y}_t]^T$. The state variable $(x_t, y_t)$ refer to the position of the target, while $(\dot{x}_t, \dot{y}_t)$ refer to the velocity of the target in Cartesian co-ordinates (2-D space is assumed here). Concatenating this state vector in each time gets a trajectory $T = [s_1, s_2, \cdots, s_n]^T$. In paper [11], the authors discretized surveillance space into a number of grid cells; each grid cell was represented by its center $\lambda_i$; and quantized the position state to the closest $\lambda_i$ at each time instant. Then stochastic context-free grammars (SCFG) were applied to convert the trajectory into a string of terminal symbols.

In this section, we use SDA to address the trajectory representation problem. Several shapes of the trajectories are considered here. A ground truth and an estimated version of the rectangle trajectory traversed by a target are shown in fig.3 while those of an arc trajectory are shown in fig.4. The horizontal and the vertical axis represent the target's position. In each case, 1000 time instances are included.

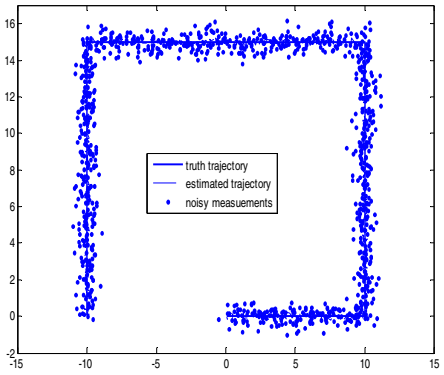


Fig.3: A rectangle trajectory, its noisy measurements and estimated output from the tracking algorithm

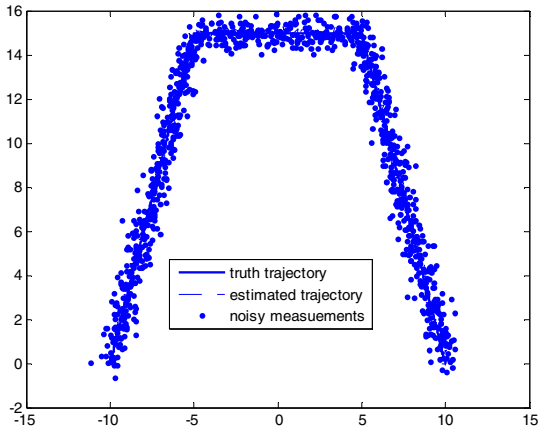


Fig.4: An arc trajectory, its noisy measurements and estimated output from the tracking algorithm

To get a corrupted input vector for the SDA, we must find a proper stochastic mapping to map the estimated trajectory from the basic level tracking algorithm to a noisy version. The basic level tracking algorithm will give variance of the estimator, which proves a hint for the stochastic mapping. In this experiment, considering the variance at each time instance, we set the mapping as:

$$\tilde{s} \sim p(\tilde{s}|s) \sim N(s, 0.09) \quad (8)$$

where $s$ refers to the estimated position of the target at some time instance, and $N(\cdot)$ denotes normal distribution. For each trajectory, 1000 corrupted samples were drawn from this distribution during the training process.

A four-layer SDA is built to extract the representation features for the trajectories, and logistic functions are taken as the activity functions of the SDA. Each element of the input vector must be normalized into the range of $[0,1]$ . Take the element of the horizontal co-ordinate as an example. It is normalized by:

$$x_{normalized} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{9}$$

where $x_{normalized}$ refers to the normalized value of the horizontal co-ordinate; $x_{\min}$ refers to the minimum horizontal co-ordinate value of the position at each time instance of the trajectory, while $x_{\max}$ refers to that of the maximum value. As there are 1000 time instances in each trajectory, the dimensionality of the input vector for the first layer of the SDA is set to be 2000. Based on the results of some preliminary experiments, the dimensionalities of the hidden variables of successive three higher layers are 1500, 300, and 40 respectively. And the dimensionality of the output layer is 4. Stochastic gradient descent algorithm is applied to optimize the parameters of the model. The initial values of the parameters are randomly sampled from the standard normal distribution. Each minibatch includes 50 samples, and a fixed learning rate of 0.02 was applied to perform 8000 weight updates. In order to visualize the results, we choose three dimensions of the output features to draw in figure 5. The circle in the figure denotes the features of the rectangle trajectory, and the triangle denotes the arc trajectory. It can be shown from figure 5 that the two different shapes of the trajectories can be properly distinguished.

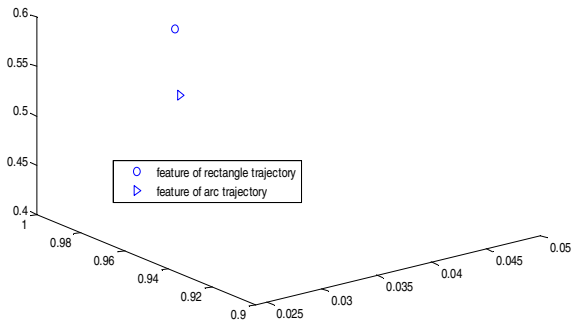


Fig.5: Output features of the two shapes of the trajectories

As the parameters of the model have been achieved, the SDA can be used to abstract representation features for some other shapes of trajectories. Figure 6 shows one semi-circle and two lines trajectories. The output features of the SDA for these three shapes of the trajectories are shown in figure 7 (We only choose three dimensionalities of the output features to draw the figure).

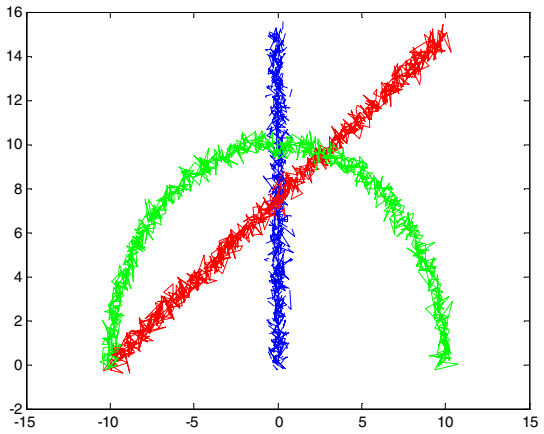

Fig. 6: three shapes of trajectories

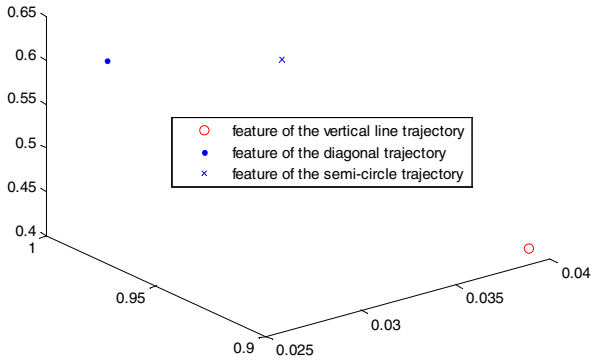


Fig.7: Output features of the tree shapes of the trajectories

As shown in figure 7, the different shapes of trajectories, which have not been seen in the training process, can be distinguished by the SDA effectively, even though no classifier is used. From the results of the preliminary experiment, we believe that the features extracted by the SDA really make sense. In the next section, the output features will be fed to SVM for abnormal detection.

# 4 Experimental Results

To verify that the feature vectors abstracted by the SDA are indeed able to be used to detect anomalous trajectories, the numeric experimental results are given in this section. The experiment is borrowed form [12], where 100 different experimental cases were considered, each one with a different number of groups of 100 normal trajectories (ranging from 1 to 10) and outliers (from 1 to 10). For each test, ten different training/test sets were created, for a total of 2000 data sets. The outliers in the test sets are drawn from the same distribution used for outliers in the corresponding training sets. The experimental data were

created according to the matlab program provided by the authors on web (http://avires.dimi.uniud.it/papers/trclust/).

In this paper, an SDA is constructed to abstract feature vectors for the trajectories. A small fraction (5%) of the data is randomly chosen to determine some super-parameters for the model. We use a stochastic mapping of a normal distribution with the mean being the trajectory data and the variance, 0.01. A three-layer SDA is applied, and the dimensionality of the two hidden variable is 25 and 14, respectively; while that of the input and output vector is 32 and 8 respectively. The activity function is logistic function, and the input vectors are normalized according to Eq. (9). Other super-parameters are setting the same as those in the preliminary experiment. Once the parameters of the SDA have been trained, the raw trajectories data are input into the SDA and the output feature vectors are fed into the single-class SVM. The results of the proposed approach in terms of false positives (%) and true positives (%) are shown in table 1 and 2, respectively. The underline indicates that the result is better than that in [11]. It can be shown from the result that using the feature vectors abstracted from the SDA improves the performance of the anomalous trajectory detection.

Table 1: False positives (%) for the proposed methods. The results have been averaged over ten runs. The underline shows that the result is better than that in [11].

| Anomalies | Clusters in the training set | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2.0 | 2.2 | 2.2 | 2.0 | 1.5 | 1.6 | 1.2 | 1.2 | 1.0 | 0.8 |
| 2 | 1.7 | 2.1 | 1.0 | 1.5 | 1.6 | 1.1 | 1.0 | 0.5 | 0.8 | 0.5 |
| 3 | 2.3 | 2.5 | 1.5 | 1.8 | 1.3 | 1.2 | 1.3 | 1.3 | 1.1 | 0.7 |
| 4 | 2.3 | 1.5 | 1.0 | 1.5 | 1.0 | 1.5 | 1.1 | 1.1 | 0.9 | 0.4 |
| 5 | 3.4 | 2.2 | 2.3 | 1.2 | 1.5 | 1.7 | 1.4 | 0.8 | 1.1 | 0.6 |
| 6 | 3.0 | 2.0 | 3.0 | 1.0 | 1.4 | 1.4 | 1.0 | 1.1 | 0.5 | 0.9 |
| 7 | 1.5 | 2.5 | 1.0 | 1.3 | 1.3 | 1.3 | 1.3 | 1.1 | 0.8 | 0.8 |
| 8 | 1.2 | 2.0 | 1.5 | 1.6 | 1.1 | 1.6 | 1.4 | 1.3 | 1.0 | 1.0 |
| 9 | 2.5 | 2.0 | 2.2 | 1.2 | 1.6 | 1.0 | 1.7 | 0.8 | 1.4 | 1.4 |
| 10 | 2.5 | 2.3 | 1.4 | 1.6 | 1.2 | 1.1 | 1.2 | 0.7 | 0.4 | 0.6 |

Table 2: True positives (%) for the proposed methods. The results have been averaged over ten runs. The underline shows that the result is better than that in [11].

| Anomalies | Clusters in the training set | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 100 | 100 | 98 | 100 | 91 | 100 | 93 | 84 | 100 | 92 |
| 2 | 100 | 98 | 100 | 100 | 100 | 100 | 97.3 | 100 | 100 | 90 |
| 3 | 100 | 100 | 100 | 99.5 | 100 | 98.7 | 90 | 92.8 | 92.1 | 94.1 |
| 4 | 100 | 100 | 100 | 98 | 95 | 96 | 95.6 | 91.5 | 96.3 | 91 |
| 5 | 100 | 96 | 100 | 100 | 99.2 | 91 | 96.2 | 90 | 87.4 | 96 |
| 6 | 100 | 97.5 | 99 | 100 | 95 | 92.1 | 92.8 | 95.8 | 92 | 84 |
| 7 | 100 | 100 | 96 | 98 | 96.5 | 91 | 97.4 | 90.8 | 90.8 | 81 |
| 8 | 100 | 96.6 | 98 | 96 | 97.2 | 98.5 | 93 | 94.6 | 95.8 | 94.7 |
| 9 | 100 | 99 | 97.8 | 98 | 97.8 | 99.1 | 91.6 | 97 | 90.2 | 82 |
| 10 | 100 | 100 | 100 | 100 | 98.9 | 98 | 98.3 | 98.5 | 94.5 | 94.8 |

# 5 Conclusion

In this paper, we propose a novel method for trajectory representation. SDA is an important algorithm in deep learning community. The corrupted vector is input into SDA, and be reconstructed as close as the uncorrupted vector: cleaning the corrupted input, or in short denoising. It is expected that performing the denoising task well requires extracting features that capture useful structure in the input distribution. The numerical experimental results show that the feature obtained from SDA can really improve the accuracy of the anomalous trajectory detection.

# Acknowledgement

This research work was supported by the Grant for State Key Program for Basic Research of China (973) (No. 2013CB329405), Foundation for Innovative Research Groups of the National Natural Science Foundation of China (61221063 )

# References

[1] Liao, T. Warren. "Clustering of time series data—a survey." *Pattern recognition* 38.11 (2005): 1857-1874.

[2] Gupta M., Jing G., te al. "Outlier Detection for Temporal Data : a survey." *Knowledge and Data Engineering, IEEE Transaction on recognition* 9.26 (2014) : 2250-2267

[3] Hu, Weiming, et al. "A system for learning statistical motion patterns."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.9 (2006): 1450-1464.

[4] Piciarelli, Claudio, and Gian Luca Foresti. "On-line trajectory clustering for anomalous events detection." *Pattern Recognition Letters* 27.15 (2006): 1835-1842.

[5] Johnson, Neil, and David Hogg. "Learning the distribution of object trajectories for event recognition." *Image and Vision computing* 14.8 (1996): 609-615.

[6] Stauffer, Chris, and W. Eric L. Grimson. "Learning patterns of activity using real-time tracking." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (2000): 747-757.

[7] Lin, Jessica, et al. "Iterative incremental clustering of time series."*Advances in Database Technology-EDBT 2004*. Springer Berlin Heidelberg, 2004. 106-122.

[8] Vlachos, Michail, et al. "A wavelet-based anytime algorithm for k-means clustering of time series." *In Proc. Workshop on Clustering High Dimensionality Data and Its Applications*. 2003.

[9] Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

[10] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507.

[11] Fanaswala, Mustafa, and Vikram Krishnamurthy. "Detection of anomalous trajectory patterns in target tracking via stochastic context-free grammars and reciprocal process models." *Selected Topics in Signal Processing, IEEE Journal of* 7.1 (2013): 76-90.

[12] Piciarelli, Claudio, Christian Micheloni, and Gian Luca Foresti. "Trajectory-based anomalous event detection." *Circuits and Systems for Video Technology, IEEE Transactions on* 18.11 (2008): 1544-1554.