Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

Outlier Detection for Time Series with Recurrent Autoencoder Ensembles

Tung Kieu , **Bin Yang*** , **Chenjuan Guo** and **Christian S. Jensen**
Department of Computer Science, Aalborg University, Denmark
{tungkvt, byang, cguo, csj}@cs.aau.dk

Abstract

We propose two solutions to outlier detection in time series based on recurrent autoencoder ensembles. The solutions exploit autoencoders built using sparsely-connected recurrent neural networks (S-RNNs). Such networks make it possible to generate multiple autoencoders with different neural network connection structures. The two solutions are ensemble frameworks, specifically an independent framework and a shared framework, both of which combine multiple S-RNN based autoencoders to enable outlier detection. This ensemble-based approach aims to reduce the effects of some autoencoders being overfitted to outliers, this way improving overall detection quality. Experiments with two real-world time series data sets, including univariate and multivariate time series, offer insight into the design properties of the proposed ensemble frameworks and demonstrate that the proposed frameworks are capable of outperforming both baselines and the state-of-the-art methods.

1 Introduction

As part of the ongoing digitalization of societal and industrial processes, many sensor-equipped devices, such as mobile phones, GPS navigators, and health care monitors, produce large volumes of time-ordered observations that form time series [Hu *et al.*, 2018]. Such time series are produced widely and are used in many application domains [Chia and Syed, 2014; Yang *et al.*, 2018; Ding *et al.*, 2016], e.g., finance, biology, transportation, and healthcare. For example, a ten-electrode electrocardiography (ECG) device monitors the tiny electrical changes on the skin of a patient over a time period. Each electrode detects and records changes in the form of a 1-dimensional time series, and the combined records from all the electrodes form a 10-dimensional time series.

Analyses of time series yield knowledge of the underlying processes that generate the time series and in turn enable us to understand those processes [Yang *et al.*, 2013]. In this paper, we focus on detecting outliers in time series, thus identifying observations that differ significantly from the remaining observations [Aggarwal, 2013]. Such outliers can offer important insights. For example, outliers in ECG time series may indicate potential heart attacks while the remaining observations represent normal physical conditions [Chia and Syed, 2014]. Further, we consider outlier detection in an *unsupervised* setting, where we do not rely on prior labels that indicate outliers. Rather, we aim at identifying outliers in unlabeled time series.

Most existing methods for outlier detection are based on similarity search [Yeh *et al.*, 2016] and density-based clustering [Breunig *et al.*, 2000]. Recently, neural network based autoencoders [Xia *et al.*, 2015; Chen *et al.*, 2017; Kieu *et al.*, 2018b] are proposed for the detection of outliers, achieving competitive accuracy. The idea is to compress original input data into a compact, hidden representation and then to reconstruct the input data from the hidden representation. Since the hidden representation is very compact, it is only possible to reconstruct representative features from the input, not the specifics of the input data, including any outliers. This way, the difference, or reconstruction errors, between the original data and the reconstructed data indicate how likely it is that observations in the data are outliers. Next, autoencoder ensembles are used to further improve the accuracy that can be achieved when using a single autoencoder that may often overfit to the original data [Chen *et al.*, 2017]. However, effective autoencoder ensembles only exist for non-sequential data, and applying them to time series data directly gives poor results (e.g., cf. the RN columns in Table 2 in Section 4.) We aim at filling this gap by proposing two recurrent neural network autoencoder ensemble frameworks to enable outlier detection in time series.

We use recurrent neural network autoencoders since they have been shown to be effective for time series learning, including for outlier detection [Kieu *et al.*, 2018b]. Autoencoder ensembles rely on the availability of multiple recurrent neural network autoencoders with different network connection structures. We propose to use *sparsely-connected recurrent neural networks* to achieve such autoencoders. In particular, we propose two ensemble frameworks, an independent framework IF and a shared framework SF that incorporate multiple sparsely-connected RNN autoencoders. Specifically, IF trains multiple autoencoders independently. In contrast, motivated by the principles of multi-task learning, SF trains multiple autoencoders jointly through a shared feature space.

*Corresponding author.

For both frameworks, we use the median of the reconstruction errors of multiple autoencoders as the final reconstruction error that quantifies how likely an observation in a time series is an outlier. As a result, the two frameworks benefit from the combination of multiple encoders and decoders.

To the best of our knowledge, this is the first proposal for using recurrent neural network autoencoder ensembles for outlier detection in time series. We make three contributions.

- We propose sparsely-connected recurrent units that enable autoencoders with different network structures.
- We propose two ensemble frameworks that combine multiple autoencoders based on sparsely-connected recurrent neural networks to advance time series outlier detections.
- We report on experiments using both univariate and multivariate time series, thus offering evidence of the effectiveness of the proposed ensemble frameworks.

2 Preliminaries

2.1 Time Series

A time series $\mathcal{T} = \langle \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_C \rangle$ is a time-ordered sequence of vectors. Each vector $\mathbf{s}_i = (s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(k)})$ represents k features of an entity at a specific time point t_i, where $1 \leq i \leq C$. In addition, we have $t_i < t_j$ when $i < j$. When $k = 1$, the time series is *univariate*; and when $k > 1$, the time series is *multivariate*.

2.2 Outlier Detection in Time Series

Given a time series $\mathcal{T} = \langle \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_C \rangle$, we aim at computing an outlier score $OS(\mathbf{s}_i)$ for each vector $\mathbf{s}_i$ such that the higher $OS(\mathbf{s}_i)$ is, the more likely it is that vector $\mathbf{s}_i$ is an outlier.

Next, we proceed to cover the basics of *autoencoders* and *autoencoder ensembles* in the context of outlier detection.

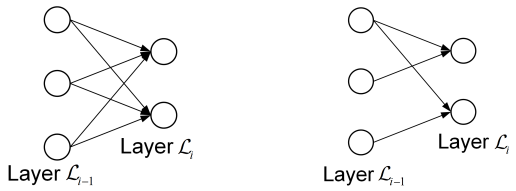
2.3 Autoencoders

A classic autoencoder is a feedforward fully-connected neural network, where the number of neurons in the input layer and the output layer are the same and where there are much fewer neurons in the hidden layers than in the input and output layers [Xia *et al.*, 2015]. An autoencoder aims to produce an output that reconstructs its input with noise removed. Since the hidden layers consist of much fewer neurons, to reconstruct the input as closely as possible, the weights in the hidden layers only capture the most representative features of the original input data and ignore the detailed specifics of the input data, such as outliers. In other words, inliers (i.e., normal data) are much easier to reconstruct than outliers. Based on the above, the larger the difference between the output (i.e., the reconstructed input) and the original input, the more likely it is that the corresponding input data is an outlier [Xia *et al.*, 2015].

Classic autoencoders based on feedforward neural networks are often used for non-sequential data. To perform outlier detection in sequential data such as time series, autoencoders based on recurrent neural networks are proposed while reusing the idea that large reconstruction errors indicate outliers [Malhotra *et al.*, 2016; Kieu *et al.*, 2018b].

2.4 Autoencoder Ensembles

Following the principles of ensemble learning, autoencoder ensembles aim to further improve the accuracy of outlier detection based on autoencoders [Chen *et al.*, 2017]. The main idea is to build a set of autoencoders and to consider the reconstruction errors from multiple autoencoders when detecting outliers. Using a set of classic, fully-connected autoencoders is not helpful since the network structures are the same across the different autoencoders. Instead, for each autoencoder, it is helpful to randomly remove some connections to obtain a sparsely-connected autoencoder (see Figure 1). Then, an autoencoder ensemble consists of multiple sparsely-connected autoencoders with different network structures, which helps reduce the variances of the overall reconstruction errors [Chen *et al.*, 2017].



(a) Fully-connected layers (b) Sparsely-connected layers

Figure 1: Fully-connected layers vs. sparsely-connected layers.

However, autoencoder ensembles are only available for non-sequential data, and these cannot be applied directly to sequential data such as time series (see a summary in Table 1). We fill this gap by proposing two autoencoder ensemble frameworks that are able to perform outlier detection in time series.

	Autoencoders	**Autoencoder Ensembles**
Non-sequential data	[Xia *et al.*, 2015; Luo and Nagarajan, 2018]	[Chen *et al.*, 2017]
Sequential data	[Malhotra *et al.*, 2016; Kieu *et al.*, 2018b]	This paper

Table 1: Outlier Detection Using Autoencoders

3 Autoencoder Ensembles For Time Series

We use recurrent neural network (RNN) autoencoders for time series because RNNs have proven to be effective for time series modeling and learning. We first discuss how to build sparsely connected RNNs, and then we propose two different ensemble frameworks that integrate multiple sparsely connected RNN autoencoders.

3.1 Sparsely-connected RNNs (S-RNNs)

A sequence-to-sequence model [Sutskever *et al.*, 2014] is often used as an autoencoder for outlier detection in time series. Such a model has an encoder and a decoder, as shown in Figure 2.

In the encoder, each vector $\mathbf{s}_t$ in time series $\mathcal{T}$ is fed into an RNN unit (shown as a square in Figure 2) to perform the

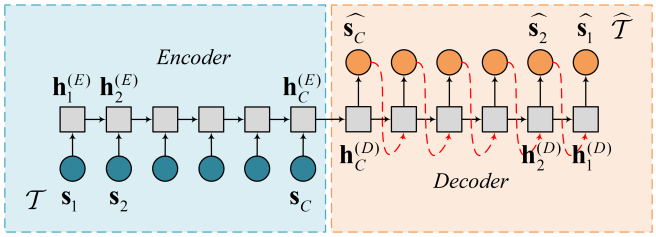


Figure 2: A Sequence-to-Sequence RNN Autoencoder

following computation.

$$\mathbf{h}_t^{(E)} = f(\mathbf{s}_t, \mathbf{h}_{t-1}^{(E)}) \tag{1}$$

Here, $\mathbf{s}_t$ is the vector at time step t in the time series and hidden state $\mathbf{h}_{t-1}^{(E)}$ is the output of the previous RNN unit at time step $t-1$ in the encoder. Next, $f(\cdot)$ is a non-linear function, ranging from the basic tanh or sigmoid to the more sophisticated Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] or Gated Recurrent Unit (GRU) [Chung *et al.*, 2014]. Based on the above, we obtain a hidden state $\mathbf{h}_t^{(E)}$ of the current RNN unit at time step t, which is then fed into the next RNN unit at time step $t+1$.

In the decoder, we reconstruct the time series in *reverse order*, i.e., $\hat{\mathcal{T}} = \langle \hat{\mathbf{s}}_C, \hat{\mathbf{s}}_{C-1}, \ldots, \hat{\mathbf{s}}_1 \rangle$. First, the last hidden state of the encoder is used as the first hidden state of the decoder. Based on the previous hidden state of the decoder $\mathbf{h}_{t+1}^{(D)}$ and the previous reconstructed vector $\hat{\mathbf{s}}_{t+1}$, we reconstruct the current vector using Equation 2 and also compute the current hidden state using Equation 3, where $f(\cdot)$ and $g(\cdot)$ are again non-linear functions.

$$\hat{\mathbf{s}}_t = g(\hat{\mathbf{s}}_{t+1}, \mathbf{h}_{t+1}^{(D)}) \tag{2}$$

EcoDco 16年icml

$$\mathbf{h}_t^{(D)} = f(\hat{\mathbf{s}}_t, \mathbf{h}_{t+1}^{(D)}) \tag{3}$$

Following the idea from existing autoencoder ensembles on non-sequential data, we aim to construct multiple autoencoders with different network structures. However, randomly removing connections between RNN units does not work in this setting because no matter which connection is removed, the RNN units become disconnected, rendering it impossible to train the network.

To contend with this challenge, we consider Recurrent Skip Connection Networks (RSCNs) that employ additional auxiliary connections among the RNN units [Wang and Tian, 2016]. In particular, each RNN unit not only considers the previous hidden state but also considers additional hidden states in the past. Formally, we have:

$$\mathbf{h}_t = \frac{f(\mathbf{s}_t, \mathbf{h}_{t-1}) + f'(\mathbf{s}_t, \mathbf{h}_{t-L})}{2} \tag{4}$$

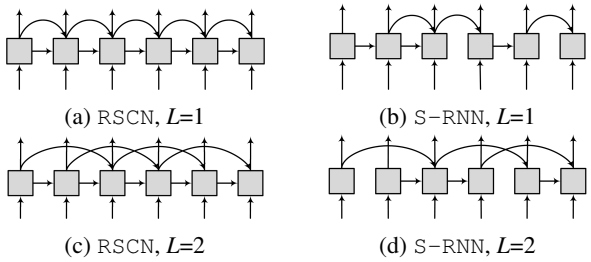
Here, we consider the previous hidden state $\mathbf{h}_{t-1}$ and the state of the hidden state L steps earlier. And we often use two different functions $f(\cdot)$ and $f'(\cdot)$. Figures 3(a) and (c) show examples of RSCNs with $L=1$ and $L=2$.

Next, based on the RSCNs, we randomly remove some connections between hidden states. Specifically, we introduce a *sparseness weight* vector $\mathbf{w}_t = (w_t^{(f)}, w_t^{(f')})$ to control which connections should be removed at each time step t, where $w_t^{(f)} \in \{0,1\}$ and $w_t^{(f')} \in \{0,1\}$. And we ensure that at least one element in $\mathbf{w}_t$ is not equal to 0, i.e., $\mathbf{w}_t = (0,1)$, $\mathbf{w}_t = (1,0)$, or $\mathbf{w}_t = (1,1)$.

Based on $\mathbf{w}_t$, we generate sparsely-connected RNNs (S-RNNs), where at each unit, the computation is as defined in Equation 5.

$$\mathbf{h}_t = \frac{f(\mathbf{s}_t, \mathbf{h}_{t-1}) \cdot w_t^{(f)} + f'(\mathbf{s}_t, \mathbf{h}_{t-L}) \cdot w_t^{(f')}}{\|\mathbf{w}_t\|_0}, \tag{5}$$

where $\|\mathbf{w}_t\|_0$ denotes the number of non-zero elements in vector $\mathbf{w}_t$. Figures 3(b) and (d) show examples of S-RNNs when $L=1$ and $L=2$.



(a) RSCN, *L*=1 (b) S-RNN, *L*=1

(c) RSCN, *L*=2 (d) S-RNN, *L*=2

Figure 3: Recurrent Skip Connection Networks vs Sparsely Connected RNNs

S-RNNs differ from RNNs using Dropout [Gal and Ghahramani, 2016]. The network connections in sparsely-connected RNNs are fixed throughout the training phase, whereas RNNs with Dropout randomly remove connections at every training epoch.

3.2 S-RNN Autoencoder Ensembles

To enable an ensemble, we build a set of S-RNN autoencoders. We then propose two different frameworks for integrating the autoencoders into ensembles.

Independent Framework

Figure 4 shows the basic, independent framework of an S-RNN autoencoder ensemble. The ensemble contains N S-RNN autoencoders that each consists of an encoder E_i and a decoder D_i, $1 \le i \le N$. Further, each autoencoder has its distinct sparseness weight vectors.

Each autoencoder in the ensemble is trained independently by minimizing the objective function $\mathcal{J}_i$ that measures the difference between the input vectors in the original time series and the reconstructed vectors, defined in Equation 6.

$$\mathcal{J}_i = \sum_{t=1}^{C} \|\mathbf{s}_t - \hat{\mathbf{s}}_t^{(D_i)}\|_2^2, \tag{6}$$

where $\hat{\mathbf{s}}_t^{(D_i)}$ denotes the reconstructed vector at time step t from decoder D_i, and $\|\cdot\|_2$ is the L2-norm of a vector.

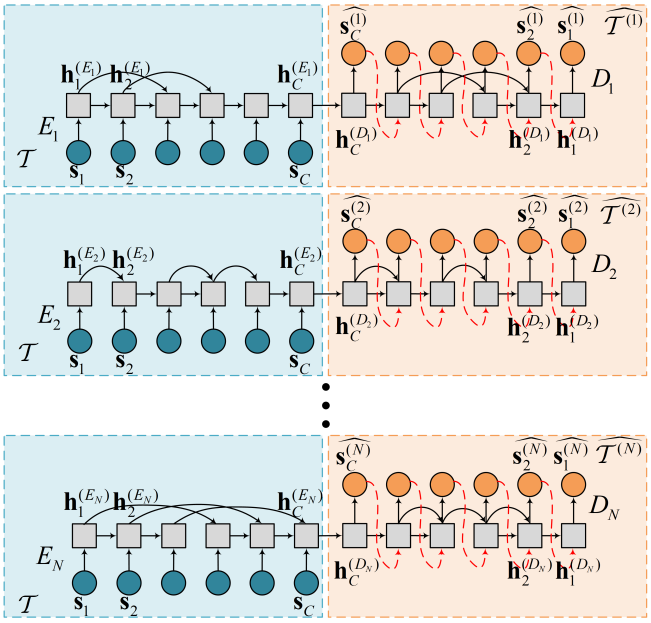


Figure 4: Independent Framework

Shared Framework

The basic framework trains different autoencoders independently, meaning that different autoencoders do not interact during the training phase. However, since all autoencoders try to reconstruct the same, original time series, it is relevant to enable interactions among the autoencoders. Motivated by the principles of multi-task learning [Long *et al.*, 2017; Cirstea *et al.*, 2018; Kieu *et al.*, 2018a], we propose a shared framework that incorporates interactions among different autoencoders. More specifically, given N tasks where each task reconstructs the original time series, we let the N tasks interact through a shared layer. The shared framework is shown in Figure 5.

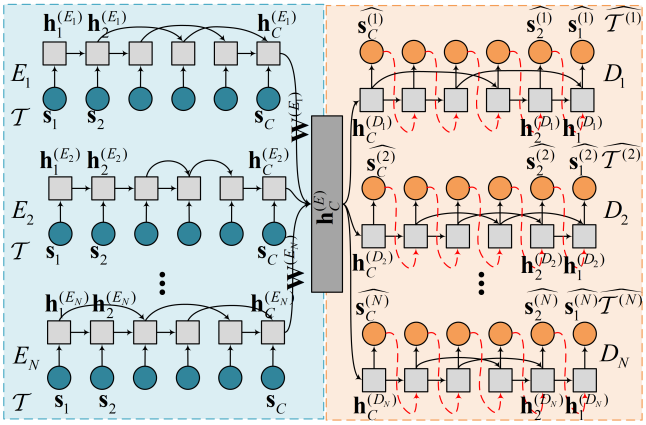


Figure 5: Shared Framework

The shared framework, shown in Figure 5, uses a shared layer, denoted as $\mathbf{h}_C^{(E)}$, to concatenate the linear combinations (using linear weight matrices $\mathbf{W}^{(E_i)}$) of all the last hidden states of all the encoders. Formally, we have:

$$\mathbf{h}_C^{(E)} = \text{concatenate}(\mathbf{h}_C^{(E_1)} \cdot \mathbf{W}^{(E_1)}, \ldots, \mathbf{h}_C^{(E_N)} \cdot \mathbf{W}^{(E_N)})$$

Each decoder D_i employs the concatenated hidden states $\mathbf{h}_C^{(E)}$ as the initial hidden state when reconstructing time series $\hat{\mathcal{T}}^{(i)} = \langle \hat{\mathbf{s}}_C^{(i)}, \ldots, \hat{\mathbf{s}}_2^{(i)}, \hat{\mathbf{s}}_1^{(i)} \rangle$. In the shared framework, all autoencoders are trained jointly by minimizing the objective function $\mathcal{J}$ that sums up the reconstruction errors of all autoencoders and an L1 regularization term on the shared hidden state:

$$\mathcal{J} = \sum_{i=1}^{N} \mathcal{J}_i + \lambda \|\mathbf{h}_C^{(E)}\|_1 \tag{7}$$

$$= \sum_{i=1}^{N} \sum_{t=1}^{C} \|\mathbf{s}_t - \hat{\mathbf{s}}_t^{(D_i)}\|_2^2 + \lambda \|\mathbf{h}_C^{(E)}\|_1 \tag{8}$$

Here, λ is a weight that controls the importance of the L1 regularization $\|\mathbf{h}_C^{(E)}\|_1$. The L1 regularization has the effect of making the shared hidden state $\mathbf{h}_C^{(E)}$ sparse. This avoids cases where some encoders overfit to the original time series and helps make the decoders robust and less affected by outliers. Hence, when autoencoders meet outliers, the difference between original time series and reconstructed time series is more pronounced.

3.3 Ensemble Outlier Scoring

Following the principles of autoencoder ensembles for non-sequential data [Chen *et al.*, 2017], we calculate the outlier score of each vector in a time series when using the ensemble frameworks. Recall that we have N autoencoders that reconstruct the original time series $\mathcal{T} = \langle \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_C \rangle$. Thus, we obtain N reconstructed time series $\hat{\mathcal{T}}^{(i)} = \langle \hat{\mathbf{s}}_C^{(i)}, \ldots, \hat{\mathbf{s}}_2^{(i)}, \hat{\mathbf{s}}_1^{(i)} \rangle$, where $1 \leq i \leq N$. For each vector s_k in the original time series $\mathcal{T}$, we compute N reconstruction errors $\{\|\mathbf{s}_k - \hat{\mathbf{s}}_k^{(1)}\|_2^2, \|\mathbf{s}_k - \hat{\mathbf{s}}_k^{(2)}\|_2^2, \ldots, \|\mathbf{s}_k - \hat{\mathbf{s}}_k^{(N)}\|_2^2\}$. We use the median of the N errors as the final outlier score of vector s_k: $OS(\mathbf{s}_k) = \text{median}(\|\mathbf{s}_k - \hat{\mathbf{s}}_k^{(1)}\|_2^2, \|\mathbf{s}_k - \hat{\mathbf{s}}_k^{(2)}\|_2^2, \ldots, \|\mathbf{s}_k - \hat{\mathbf{s}}_k^{(N)}\|_2^2)$. Using median instead of mean reduces the influence of the reconstruction errors from the autoencoders that overfit to the original time series.

4 Experiments

4.1 Experimental Setup

Data Sets

We use two real-world time series repositories, the univariate time series repository Numenta Anomaly Benchmark (*NAB*)* and the multivariate time series repository Electrocardiography (*ECG*)†. *NAB* comprises six sets of time series from different domains, where each set has ca. 10 univariate time series and each time series contains from 5,000 to 20,000 observations.

*https://github.com/numenta/NAB

†http://www.cs.ucr.edu/ eamonn/discords/ECG_data.zip/

ECG comprises seven 3-dimensional time series from seven patients, where each time series has 3,750 to 5,400 observations. For both repositories, ground truth labels of outlier observations are available. However, consistent with the unsupervised setting, we do not use these labels for training, but only use them for evaluating accuracy.

Existing Solutions

We compare the proposed independent and shared ensemble frameworks (IF and SF) with seven competing solutions–(1) Local Outlier Factor (LOF) [Breunig *et al.*, 2000], a well-known density-based outlier detection method; (2) One-class Support Vector Machines (SVM) [Manevitz and Yousef, 2001], a kernel-based method, (3) Isolation Forest (ISF) [Liu *et al.*, 2008], which is a randomized clustering forest, (4) Matrix Profile I (MP) [Yeh *et al.*, 2016], which is the state-of-the-art similarity-based outlier detection method, (5) RandNet (RN) [Chen *et al.*, 2017], which uses the state-of-the-art feedforward autoencoder ensembles for non-sequential data, (6) Convolutional Neural Networks based Autoencoder (CNN) [Kieu *et al.*, 2018b], which treats time series as images and employ a CNN autoencoder to reconstruct the image, and (7) an LSTM based Autoencoder(LSTM) [Malhotra *et al.*, 2016], which is the state-of-the-art deep learning outlier detection method for time series. In addition, we replace the LSTM units in method LSTM with RSCN units [Wang and Tian, 2016] to study the effect of using a single RSCN autoencoder (RSCN) versus the use of multiple S-RNN autoencoders in IF and SF. Methods LOF, SVM, and ISF were originally proposed for non-sequential data but can also be applied to sequential data with competitive accuracy [Aggarwal, 2013]. This is why we include them in the experiments.

Implementation Details

All algorithms are implemented in Python 3.5. Methods IF and SF and the deep learning methods, i.e., CNN, LSTM, and RSCN, are implemented using TensorFlow 1.4.0, while the remaining methods, i.e., LOF, SVM, ISF, and MP are implemented using Scikit-learn 1.19. The source code is available at https://github.com/tungk/OED. Experiments are performed on a Linux workstation with dual 12-core Xeon E5 CPUs, 64 GB RAM, and 2 K40M GPUs.

Hyperparameters Settings

For all deep learning based methods, we use Adadelta [Zeiler, 2012] as the optimizer, and we set their learning rates to 10^{-3}. For the proposed ensemble frameworks, we use an LSTM unit and tanh as the functions f and f' in Equation 5; we set the number of hidden LSTM units to 8; we set the default number of autoencoders N to 40, and we also study the effect of varying N from 10 to 40; and we set λ to 0.005. We randomly vary the skip connection jump step size L from 1 to 10, and we randomly choose the sparse weight vector $\mathbf{w}_t$. For MP, we set the pattern size to 10. For RN and all the other baselines, we follow the settings used by [Chen *et al.*, 2017] and [Kieu *et al.*, 2018b], respectively.

Evaluation Metrics

A typical approach to decide which vectors in a time series are outliers is to set a threshold and to consider the vectors whose outlier scores exceed the threshold as outliers. However, setting the threshold is non-trivial and often requires human experts or prior knowledge. Instead, following the evaluation metrics used for evaluating autoencoder ensembles on non-sequential data [Chen *et al.*, 2017], we employ two metrics that consider all possible thresholds—Area Under the Curve of Precision-Recall (*PR-AUC*) [Sammut and Webb, 2017] and Area Under the Curve of Receiver Operating Characteristic (*ROC-AUC*) [Sammut and Webb, 2017]. In other words, the two metrics do not depend on a specific threshold. Rather, they reflect the full trade-off among true positives, true negatives, false positives, and false negatives. Higher *PR-AUC* and *ROC-AUC* values indicate higher accuracy.

4.2 Experimental Results

Overall Accuracy

We report *PR-AUC* and *ROC-AUC* for all methods in Table 2. First, the deep learning based methods and the non-deep learning methods show similar results w.r.t. *PR-AUC*. However, the deep learning based methods perform better when using *ROC-AUC* in most cases. This indicates that the deep learning based methods have lower false positive rates and higher true negative rates, which is highly desired for outlier detection. Second, SF and IF outperform RN, indicating that S-RNN autoencoder ensembles are better suited for sequential data than are feedforward autoencoder ensembles. Third, the two proposed ensemble methods most often outperform the individual methods. In particular, IF and SF outperform the other methods in most cases on both univariate and multivariate time series. Specifically, IF performs the best in 4 out of 13 cases, and SF is best in 6 out of 13 cases for both evaluation metrics, indicating that the proposed ensemble frameworks are capable of improving effectiveness and robustness. Fourth, SF performs better than IF on average—when considering average *PR-AUC* and *ROC-AUC*, SF is best and IF is often second best on both univariate and multivariate time series. This indicates that multi-task learning can be applied in an ensemble framework to further enhance accuracy.

Effect of N

We study the effect of the number of autoencoders N in the two ensemble frameworks. In particular, we vary N among 10, 20, 30, and 40. Figures 6 and 7 report *PR-AUC* and *ROC-AUC*, respectively. Due to the space limitation, we only report results on 4 *NAB* datasets and 5 *ECG* datasets. As N increases, both ensemble frameworks achieve better results, i.e., higher *PR-AUC* and higher *ROC-AUC* scores. This suggests that having more autoencoders with different structures in an ensemble yields better accuracy.

IF vs. SF

Finally, we study the difference between IF and SF. In all previous experiments, IF and SF use randomly generated S-RNNs, meaning that the S-RNNs used in IF and SF may have different structures. In the final set of experiments, we make sure that both IF and SF use the same set of S-RNNs. We first generate 40 S-RNNs based autoencoders and then build IF and SF on top of the 40 autoencoders. Due to the space limitation, we only report the results on the remaining 2 *NAB* datasets and 2 *ECG* datasets.

		PR-AUC										*ROC-AUC*									
	Data Set	**SVM**	**LOF**	**ISF**	**MP**	**RN**	**CNN**	**LSTM**	**RSCN**	**IF**	**SF**	**SVM**	**LOF**	**ISF**	**MP**	**RN**	**CNN**	**LSTM**	**RSCN**	**IF**	**SF**
NAB, Univariate	**Cloudwatch**	0.908	0.904	0.909	0.892	0.886	0.889	0.886	0.895	0.885	**0.912**	0.566	0.516	0.559	0.563	0.569	0.632	0.662	0.634	**0.675**	0.667
	Anomaly	0.902	0.902	0.911	0.894	0.902	**0.926**	0.904	0.915	0.912	0.917	0.551	0.522	0.627	0.613	0.629	0.657	0.753	0.755	0.761	**0.777**
	Traffic	0.908	0.905	0.914	0.913	0.895	0.919	0.859	0.865	**0.922**	0.914	0.551	0.584	0.566	0.548	0.568	0.608	0.625	0.667	0.645	**0.689**
	Tweets	0.904	0.902	0.911	0.901	0.898	0.907	0.876	0.893	**0.915**	0.907	0.538	0.527	0.557	0.533	0.564	0.571	0.554	0.599	0.608	**0.653**
	AdExchange	0.898	0.897	0.902	0.897	0.891	0.881	0.882	0.893	0.893	**0.903**	0.551	0.513	0.539	0.567	0.565	0.602	0.574	0.603	**0.691**	0.646
	KnownCause	0.929	0.905	0.915	0.893	0.889	0.891	0.872	0.916	0.916	**0.987**	0.639	0.517	**0.657**	0.584	0.612	0.613	0.642	0.609	0.638	0.621
	Average	0.908	0.903	0.910	0.898	0.894	0.902	0.880	0.896	0.907	**0.923**	0.566	0.530	0.584	0.568	0.585	0.614	0.635	0.645	0.670	**0.676**
ECG, Multivariate	**chf01**	0.912	0.923	0.886	0.866	0.943	0.967	0.904	**0.971**	0.961	0.958	0.629	0.521	0.508	**0.815**	0.628	0.714	0.646	0.702	0.681	0.689
	chf13	**0.977**	0.964	0.975	0.936	0.941	0.944	0.938	0.966	0.963	0.967	0.618	0.577	0.658	0.644	0.661	0.642	0.581	0.562	**0.823**	0.803
	ltstdb240	0.984	0.963	0.991	0.906	0.932	0.980	0.947	0.973	0.987	**0.995**	0.678	0.530	0.798	0.875	0.727	0.747	0.651	0.587	0.753	**0.879**
	ltstdb43	0.982	0.959	0.980	0.964	0.962	0.974	0.956	0.966	0.982	**0.985**	0.617	0.531	0.612	0.512	0.558	0.561	0.589	0.581	0.709	**0.717**
	mitdb180	0.953	0.969	0.944	0.958	0.955	0.985	0.974	0.973	**0.988**	0.980	**0.720**	0.539	0.694	0.654	0.673	0.670	0.610	0.580	0.678	0.694
	stdb308	0.879	0.955	0.897	0.902	0.911	0.978	0.977	0.981	0.983	**0.987**	0.821	0.546	0.722	0.820	0.786	0.639	0.735	0.842	**0.856**	0.838
	xmitdb108	0.949	0.934	0.901	0.888	0.949	0.963	0.862	0.845	**0.964**	0.957	0.516	0.539	0.653	0.728	0.603	0.796	0.816	0.813	0.785	**0.822**
	Average	0.948	0.952	0.939	0.917	0.942	0.970	0.937	0.954	0.975	**0.976**	0.657	0.540	0.664	0.721	0.662	0.681	0.661	0.667	0.755	**0.777**

Table 2: Overall Accuracy

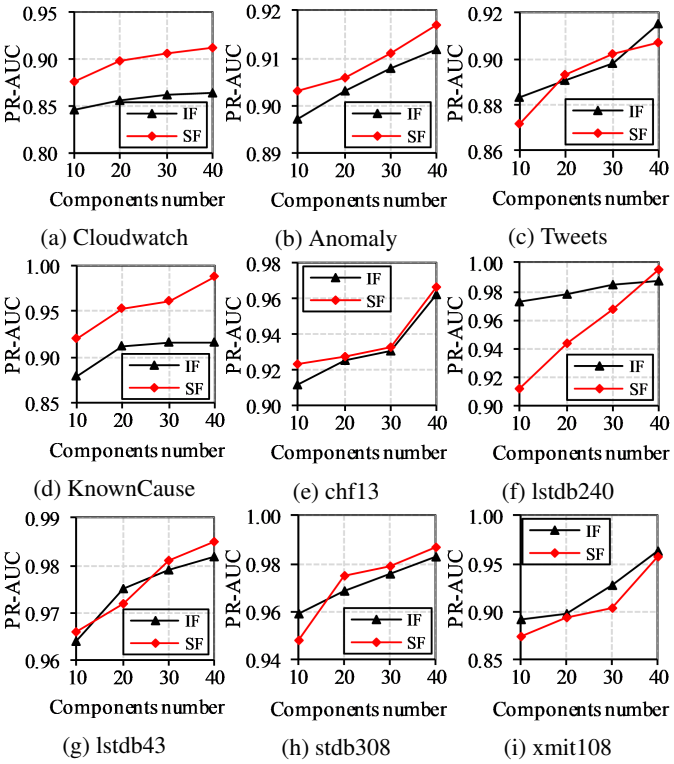


Figure 6: Effect of N, *PR-AUC*

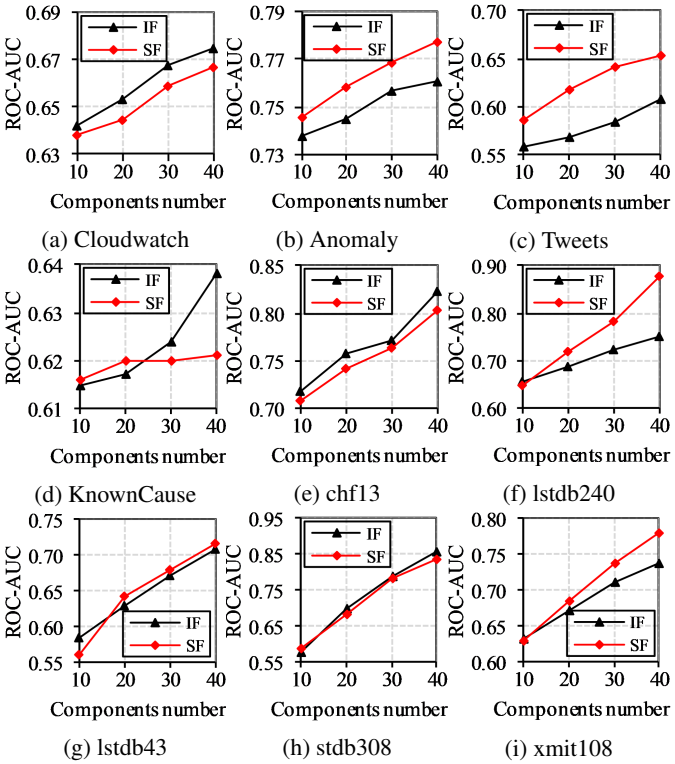


Figure 7: Effect of N, *ROC-AUC*

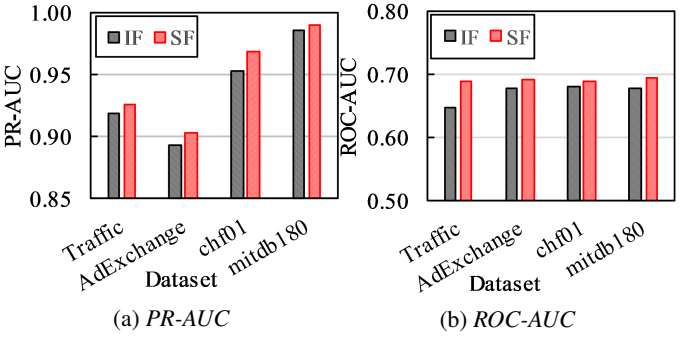


Figure 8: `IF` vs. `SF` with the Same Set of Autoencoders

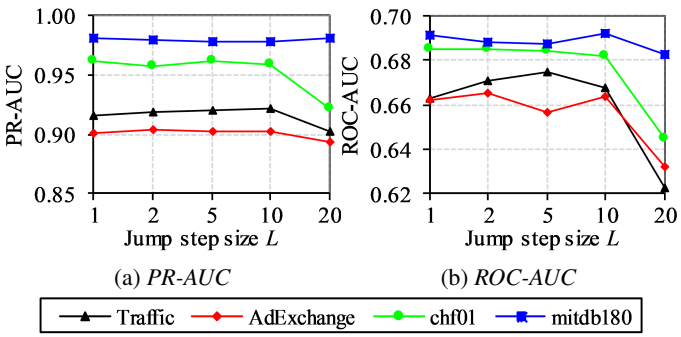


Figure 9: Effect of L, *PR-AUC* and *ROC-AUC*

Figure 8 shows that SF outperforms IF when both framework have the same set of autoencoders. This suggests that learning multiple autoencoders in a shared multi-task manner is indeed helpful. However, a disadvantage of SF is its high memory consumption that occurs because it is necessary to train many autoencoders together: the shared layer needs to combine the last states of all autoencoders before proceeding to the decoding phase. In contrast, we can train different autoencoders in IF in parallel and then calculate outlier scores at the end.

Effect of L

We study the effect of varying L when producing recurrent skip connection networks. We only report results on SF since IF yields similar results. We vary L among 1, 2, 5, 10, and 20. Figures 9(a) and (b) show *PR-AUC* and *ROC-AUC*, respectively. We only report results on the remaining 2 *NAB* datasets and 2 *ECG* datasets. We observe that from $L = 1$ to $L = 10$, L does not affect the accuracy significantly, indicating that the ensemble frameworks are insensitive to small jump steps. When $L = 20$, the accuracy starts to decrease, suggesting that (1) important temporal information was lost when combining large jump steps with sparse connections and that (2) the model was forced to employ obsolete historical information that adversely affects the accuracy.

5 Related Work

Outlier Detection using Autoencoders

Deep learning based outlier detection methods have been applied widely in many domains, e.g., computer vision [Zhao *et al.*, 2017], network analysis [Luo and Nagarajan, 2018], and robotics [Park *et al.*, 2016]. Autoencoders are at the core of these methods. In addition, studies exist that modify classic autoencoders to achieve denoising autoencoders [Tagawa *et al.*, 2014] and variational autoencoders [Liao *et al.*, 2018]. Our study falls into the category of outlier detection in time series using deep learning, where the fundamental problems are to learn a representation of the time series in a smaller vector space and to reconstruct the original time series from that vector space [Malhotra *et al.*, 2016; Kieu *et al.*, 2018b]. However, existing studies are limited to a single recurrent neural network autoencoder. In contrast, we propose two novel ensemble frameworks that exploit multiple recurrent neural network autoencoders for outlier detection in time series. The experimental results provide evidence that the ensemble frameworks are effective and are capable of outperforming existing methods that use only a single autoencoder.

Deep Ensemble Models

Ensemble models with deep learning have been applied to different supervised learning tasks, including classification [Akhtar *et al.*, 2017] and regression [Lee *et al.*, 2017]. Two popular approaches are bagging, where multiple models are trained with the same structure using randomly drawn subsets of training data [Guo *et al.*, 2015], and stacking, where multiple models are trained with different structures using the entire training data [Deng and Platt, 2014]. However, only a few deep ensembles exist for unsupervised learning, specifically for outlier detection [Aggarwal and Sathe, 2017]. An ensemble outlier detection method is proposed [Chen *et al.*, 2017] that works only for non-sequential data. This paper is the first to propose autoencoder ensembles for unsupervised outlier detection in time series. The proposed ensembles adopt the stacking approach.

6 Conclusion and Outlook

We propose two autoencoder ensemble frameworks based on sparsely-connected recurrent neural networks for unsupervised outlier detection in time series. One of the frameworks trains multiple autoencoders independently while the other framework trains multiple autoencoders jointly in a multi-task learning fashion. Experimental studies show that the proposed autoencoder ensembles are effective and outperform baselines and state-of-the-art methods.

In future work, it is of interest to study different autoencoder structures, e.g., denoising autoencoders [Vincent *et al.*, 2008] and the use of advanced deep neural network structures, e.g., embedding [Chen *et al.*, 2016] and attention [Luong *et al.*, 2015], to further improve accuracy.

Acknowledgments

This research was supported in part by a grant from the Obel Family Foundation, a grant from Independent Research Fund Denmark, and by the DiCyPS center, funded by Innovation Fund Denmark.

References

[Aggarwal and Sathe, 2017] Charu C. Aggarwal and Saket Sathe. *Outlier Ensembles - An Introduction*. Springer, 2017.

[Aggarwal, 2013] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[Akhtar *et al.*, 2017] Md. Shad Akhtar, Abhishek Kumar, Deepanway Ghosal, Asif Ekbal, and Pushpak Bhattacharyya. A multilayer perceptron based ensemble technique for fine-grained financial sentiment analysis. In *EMNLP*, pages 540–546, 2017.

[Breunig *et al.*, 2000] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.

[Chen *et al.*, 2016] Ting Chen, Lu An Tang, Yizhou Sun, Zhengzhang Chen, and Kai Zhang. Entity embedding-based anomaly detection for heterogeneous categorical events. In *IJCAI*, pages 1396–1403, 2016.

[Chen *et al.*, 2017] Jinghui Chen, Saket Sathe, Charu C. Aggarwal, and Deepak S. Turaga. Outlier detection with autoencoder ensembles. In *SDM*, pages 90–98, 2017.

[Chia and Syed, 2014] Chih-Chun Chia and Zeeshan Syed. Scalable noise mining in long-term electrocardiographic time series to predict death following heart attacks. In *KDD*, pages 125–134, 2014.

[Chung *et al.*, 2014] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[Cirstea *et al.*, 2018] Razvan-Gabriel Cirstea, Darius-Valer Micu, Gabriel-Marcel Muresan, Chenjuan Guo, and Bin Yang. Correlated time series forecasting using multi-task deep neural networks. In *CIKM*, pages 1527–1530, 2018.

[Deng and Platt, 2014] Li Deng and John C. Platt. Ensemble deep learning for speech recognition. In *INTERSPEECH*, pages 1915–1919, 2014.

[Ding *et al.*, 2016] Zhiming Ding, Bin Yang, Yuanying Chi, and Limin Guo. Enabling smart transportation systems: A parallel spatio-temporal database approach. *IEEE Trans. Computers*, 65(5):1377–1391, 2016.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, pages 1019–1027, 2016.

[Guo *et al.*, 2015] Haiyun Guo, Jinqiao Wang, and Hanqing Lu. Learning deep compact descriptor with bagging auto-encoders for object retrieval. In *ICIP*, pages 3175–3179, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2018] Jilin Hu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. Risk-aware path selection with time-varying, uncertain travel costs: a time series approach. *VLDB J.*, 27(2):179–200, 2018.

[Kieu *et al.*, 2018a] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. Distinguishing trajectories from different drivers using incompletely labeled trajectories. In *CIKM*, pages 863–872, 2018.

[Kieu *et al.*, 2018b] Tung Kieu, Bin Yang, and Christian S. Jensen. Outlier detection for multidimensional time series using deep neural networks. In *MDM*, pages 125–134, 2018.

[Lee *et al.*, 2017] Inwoong Lee, Doyoung Kim, Seoungyoon Kang, and Sanghoon Lee. Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks. In *ICCV*, pages 1012–1020, 2017.

[Liao *et al.*, 2018] Weixian Liao, Yifan Guo, Xuhui Chen, and Pan Li. A unified unsupervised Gaussian mixture variational autoencoder for high dimensional outlier detection. In *BigData*, pages 1208–1217, 2018.

[Liu *et al.*, 2008] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *ICDM*, pages 413–422, 2008.

[Long *et al.*, 2017] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S. Yu. Learning multiple tasks with multilinear relationship networks. In *NIPS*, pages 1593–1602, 2017.

[Luo and Nagarajan, 2018] Tie Luo and Sai G. Nagarajan. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. In *ICC*, pages 1–6, 2018.

[Luong *et al.*, 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421, 2015.

[Malhotra *et al.*, 2016] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148, 2016.

[Manevitz and Yousef, 2001] Larry M. Manevitz and Malik Yousef. One-class SVMs for document classification. *JMLR*, 2:139–154, 2001.

[Park *et al.*, 2016] Daehyung Park, Zackory M. Erickson, Tapomayukh Bhattacharjee, and Charles C. Kemp. Multimodal execution monitoring for anomaly detection during robot manipulation. In *ICRA*, pages 407–414, 2016.

[Sammut and Webb, 2017] Claude Sammut and Geoffrey I. Webb, editors. *Encyclopedia of Machine Learning and Data Mining*. Springer, 2017.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[Tagawa *et al.*, 2014] Takaaki Tagawa, Yukihiro Tadokoro, and Takehisa Yairi. Structured denoising autoencoder for fault detection and analysis. In *ACML*, 2014.

[Vincent *et al.*, 2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[Wang and Tian, 2016] Yiren Wang and Fei Tian. Recurrent residual learning for sequence classification. In *EMNLP*, pages 938–943, 2016.

[Xia *et al.*, 2015] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *ICCV*, pages 1511–1519, 2015.

[Yang *et al.*, 2013] Bin Yang, Chenjuan Guo, and Christian S. Jensen. Travel cost inference from sparse, spatio-temporally correlated time series using markov models. *PVLDB*, 6(9):769–780, 2013.

[Yang *et al.*, 2018] Bin Yang, Jian Dai, Chenjuan Guo, Christian S. Jensen, and Jilin Hu. PACE: a path-centric paradigm for stochastic path finding. *VLDB J.*, 27(2):153–178, 2018.

[Yeh *et al.*, 2016] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *ICDM*, pages 1317–1322, 2016.

[Zeiler, 2012] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[Zhao *et al.*, 2017] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *ACM MM*, pages 1933–1941, 2017.