TCAE: Temporal Convolutional Autoencoders for Time Series Anomaly Detection

Jinuk Park*
Korea Electronics Technology Institute
KETI
Seoul, Republic of Korea
jinuk.park@keti.re.kr

Yongju Park
Korea Electronics Technology Institute
KETI
Seoul, Republic of Korea
suede8247@keti.re.kr

Chang-il Kim
Korea Electronics Technology Institute
KETI
Seoul, Republic of Korea
mykwor2468@keti.re.kr

***Abstract*—**Prevalent recurrent autoencoders for time series anomaly detection often fail to model time series since they have information bottlenecks from the fixed-length latent vectors. In this paper, we propose a conceptually simple yet experimentally effective time series anomaly detection framework called temporal convolutional autoencoder (TCAE). Our model imposes dilated causal convolutional neural networks to capture temporal features while avoiding inefficient recurrent models. Also, we utilize bypassing residual connections in encoded vectors to enhance the temporal features and train the entire model efficiently. Extensive evaluation on several real-world datasets demonstrates that the proposed method outperforms strong anomaly detection baselines.

Keywords—time series; anomaly detection; neural networks; non-autoregressive decoding; residual connection

I. Introduction

Time series anomaly detection refers to distinguish unexpected deviation that differs significantly from the other observations. It is crucial to detect abnormal events in modern applications, ranging from industrial processes to healthcare systems [1]. For example, detecting anomalous intrusion can prevent potential dangers in many digitalized applications as smart factories, big data centers, and environmental systems [2, 3]. Also, monitoring time series records, including electrocardiography signals, are directly connected to patients, and detecting deviations in those signals can alert critical issues in advance of critical problems [4].

Recent deep learning techniques pave and advance the anomaly detection methods, as learning complex patterns and non-linear features in time series through flexible representation inside neural methods [5]. Due to the lack of labeled anomalies in real-world applications, the unsupervised learning scheme is a prevalent learning method to capture normal behaviors using neural networks. In particular, reconstruction-based methods encode the given time series into the low-dimensional latent vectors that include core information to reconstruct the original time series [6, 7]. Thus, the models can detect unexpected deviation by calculating errors between decoded and original time series. The main assumption behind this approach is that the low-dimensional and compact codes are forced to contain information related to the normal behaviors, not anomalous effects.

*Corresponding author: jinuk.park@keti.re.kr

This research was supported by a grant(RS-2022-00142239) from "A Study on the development of road infrastructure innovation technology for future environment change" Program funded by Ministry of Land, Infrastructure and Transport of Korean government

Most existing anomaly detection methods based on neural networks utilize recurrent neural networks (RNNs) owing to the structural advantages for modeling sequential data. Especially, autoencoder architectures comprising two different RNNs for encoder and decoder are considered a natural starting point for the time series reconstruction method [8]. Indeed, proposed studies based on RNN autoencoders show significant advances in detecting anomalies on real-world benchmarks [6, 9]. Previous studies have been developed in the context of improving temporal dependencies in the recurrent models using skip connections or ensemble methods. [10] proposed ensemble of encoders with skip connections to encode longer sequences using several skip lengths. Also, [11] introduced multiple RNN-based decoders with different resolutions in an ensemble manner. However, information inside canonical recurrent models is prone to fade as the sequence gets longer. Also, existing sequence-to-sequence architectures rely on the fixed-length vectors between the encoders and decoders, which causes severe information bottlenecks.

In this paper, we present a novel temporal convolutional autoencoder (TCAE) for time series anomaly detection. TCAE comprises convolutional neural networks (CNNs) to encode and reconstruct temporal features, avoiding inefficient recurrent models. Recent studies show strong evidence that simple CNNs outperform RNNs on sequential modeling tasks including representation learning. Our model imposes dilated kernels to enlarge the receptive field for long sequences. Also, our model learns latent vectors at each time step to address the information bottleneck from the fixed-length vectors between encoders and decoders. Furthermore, we construct bypass connections for encoded vectors to enhance the temporal features and train the stacked model efficiently. The main contributions in this paper can be summarized as follows:

- We propose a novel temporal convolutional autoencoder, named TCAE, for time series anomaly detection. Our model captures temporal dependencies and internal relationships between time series based on convolution operations.

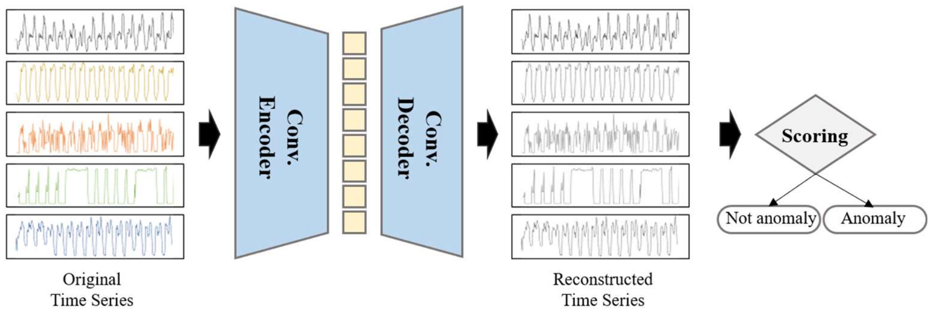


Fig. 1. Overview of Temporal Convolutional Autoencoder (TCAE)

- We prove that TCAE can avoid information bottlenecks from fixed-length vectors by capturing temporal features at each step with residual connections. Subsequently, our model reconstructs time series in a non-autoregressive manner based on the latent vectors while keeping causality by causal convolutions.
- We conduct extensive experiments on real-world datasets for time series anomaly detection. Our results demonstrate that TCAE detects anomalies more accurately than solid baselines.

II. Related Work

In this section, we briefly review previous studies, including reconstruction-based approaches for time series anomaly detection. Also, we discuss related studies using convolutional neural networks in the time series domain.

A. Time Series Anomaly Detection using Autoencoders

The major challenge in anomaly detection tasks is the unbalanced characteristics of time series, i.e., the lack of labeled anomalies [5]. Accordingly, direct supervision from the small amount of anomalies is not a viable option. Instead, most anomaly detection methods are trained based on only normal points in an unsupervised manner. For this reason, reconstruction-based methods are prevalent to identify deviations in time series.

Recently, anomaly detection based on neural methods has become a field of interest, owing to the expressive and flexible ability to represent information. In particular, autoencoders (AEs) [12] are widely used models to detect anomalies by reconstructing each datapoint. Extending AEs, one of the well-known approaches to model sequential data is a sequence-to-sequence model [13]. The sequence-to-sequence approaches generally have two distinct models called encoders and decoders. These modules serve different roles; encoders learn hidden representation (codes) in a low-dimensional vector space, and decoders reconstruct the original datapoint from the codes.

The sequence-to-sequence approaches using RNNs have become favored choices due to recurrent behaviors of sequential data. Several existing studies adopt these approaches that encode time series using a long short-term memory (LSTM) into low dimensional space and then decode them in time-reverse order to alleviate the time lag problem in RNNs [6, 9]. There have been several attempts to enhance temporal dependencies and mitigate the inherent long-term dependency problems in RNNs [14, 9]. For instance, [10] utilized several sparsely connected RNNs with different skip lengths as encoders. Similarly, [11] introduced multiple RNN decoders with various resolutions to reconstruct time series.

However, previous sequence-to-sequence models using RNNs are inevitably limited to having the fixed-length codes in the middle between encoder and decoder. These bottlenecks can cause loss of temporal features for sequential data since long sequence are compressed into a single low-dimensional vector. Attention mechanism has been studied to deliver information from encoders to decoders efficiently [15, 16]. Nevertheless, RNNs are being replaced by models that do not have long-term dependency problems such as CNNs due to their fundamental flaws [17].

B. Convolutional Networks for Time Series

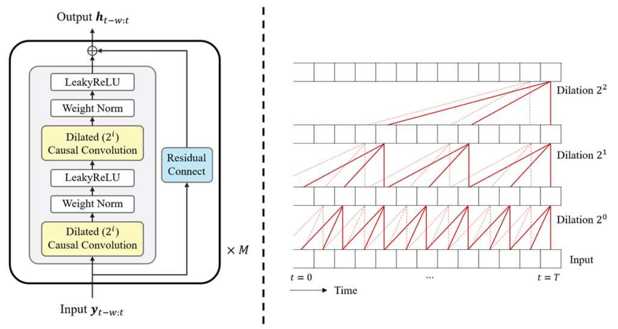
Similar to computer vision which is dominated by CNNs, deep convolutional neural networks have shown competitive performance in sequential modeling. In recent studies on time series domain, CNNs have been explosively studied owing to their powerful feature extraction performance using local kernels. Indeed, CNN-based models have achieved prominence in various time series problems [18, 19, 20].

Moreover, there is convincing evidence that dilated convolutional operations can further enhance the performance of sequence modeling such as forecasting, generation, and representation learning, even outperforming sequence-to-sequence models [21, 22, 23]. This is due to the fact that dilated convolutions with exponentially increasing kernels can efficiently enlarge receptive fields for long sequences.

III. Proposed Method

In this paper, we introduce a simple yet effective autoencoder model for time series anomaly detection, a temporal convolutional autoencoder (TCAE). As shown in Fig. 1, the main approaches our model are 1) to leverage compressed codes to reconstruct at each time step, and 2) to alleviate inefficient autoregressive decoding steps in RNN-based decoders.

Since we aim to reconstruct the given time series, not to generate future values, our model can safely utilize the entire latent vectors at all time steps instead of a limited fixed-length vector. Also, our model can reconstruct time series without



(a) Dilated CNN Architecture (b) Example of Dilated Causal Convolutions

Fig. 2. Overview of Dilated Causal CNN

autoregressive decoding while maintaining causality in dilated causal CNN.

A. Problem Formulation

We use a bold capital letter to represent a set of time series, $\mathbf{Y} \in \mathbb{R}^{N \times T}$ for N number of time series with Tnumber of time steps. Each vector $\boldsymbol{y}_t \in \mathbb{R}^N$represents all variables at time step t. Also, given a window size w, we use $\mathbf{Y}_w$ for train and test the model, i.e., $\mathbf{Y}_w = \{\boldsymbol{y}_t, \boldsymbol{y}_{t+1}, \dots, \boldsymbol{y}_{t+w-1}\}$.

The goal of time series anomaly detection is to identify anomalous time steps in test sets. Following previous anomaly detection methods, we train our model with solely normal data and inference on labeled test sets to compare performance. We mark label as 1 for anomaly and 0 otherwise.

B. Dilated Causal CNN

For encoder and decoder architecture in TCAE, we employ a dilated causal convolutional network (DC-CNN) [21]. The DC-CNN imposes exponentially growing dilation filters at each 1D convolutional layer. Also, to keep causality at time step t, the filter only convolves the time step t and the earlier inputs. With this causal convolution, the network does not learn from future values at the given time step. The Fig. 2 (b) illustrates the example of a stack of dilated causal convolutional layers.

We use two dilated convolutional layers with weight normalization, LeakyReLU as non-linear activation, and residual connections to form a convolutional block. Based on this core block, we stack M convolutional blocks for a dilated causal CNN, as depicted in Fig. 2 (a). Since each block has two dilated convolutional layers, the network can see exponentially longer past observations with stacked blocks. For simplicity, we denote the dilated causal CNN as $\mathcal{T}(\cdot)$.

C. Encoder

Autoencoders stand on the idea that encoders compress the vectors into the codes, and decoders reconstruct the original vectors from the codes. Based on this simple solid concept, TCAE aims to encode time series into T number of codes and decode them into the original series in a non-autoregressive manner.

To reconstruct the given time series, we first generate non-overlapping subsets $\mathbf{Y}_w$ with window size w. Then, DC-CNN encoder extracts feature maps from $\mathbf{Y}_w$ using k kernels to capture temporal dependencies and relationships between variables. Note that we use paddings to match the extracted length of feature maps to the original $\mathbf{Y}_w$. Also, we add the residual connection to the feature maps with linear transformation $f(\cdot)$ of $\mathbf{Y}_w$. Finally, we employ a linear transformation to feature maps to generate compressed codes. The entire process can be formulated as follows:

$$\mathbf{E}_w = \mathcal{T}_{encoder}(\mathbf{Y}_w) \tag{1}$$

$$\boldsymbol{z}_t = \boldsymbol{W}_z\big(\boldsymbol{e}_t + f(\boldsymbol{y}_t)\big) \tag{2}$$

where $\mathbf{E} \in \mathbb{R}^{k \times w}$ and $\boldsymbol{e}_t$ denote extracted feature maps from DC-CNN, and a feature map at time step t respectively. k is the number of kernels in DC-CNN. Also, $\boldsymbol{z}_t$ and $\mathbf{W}_z$ denote a latent vector at time steep t and free parameters in the linear transformation in (2).

D. Decoder

Since TCAE targets reconstructing the given time series, not generating future values, decoders can safely utilize the entire latent vectors at all time steps instead of a limited fixed-length vector. In particular, encoders have already extracted necessary information for normal behavior using the historical lookback (i.e., the past observations) at each time step t. Thus, the decoded output at t would not critically affect reconstructing the normal value at the $t + 1$ time step. Hence the model does not consider the previous outputs to reconstruct the current time step.

Instead, we feed the latent vector at time step t, and those at the past time steps into the DC-CNN in the decoder to obtain the hidden representation $\mathbf{D}$. In this way, our model can reconstruct time series without autoregressive decoding while maintaining causality in dilated causal CNN.

$$\mathbf{D}_w = \mathcal{T}_{decoder}(\mathbf{Z}_w) \tag{3}$$

$$\boldsymbol{h}_t = \mathrm{LeakyReLU}\big(\mathrm{BatchNorm}(\boldsymbol{d}_t)\big) \tag{4}$$

Where $\mathbf{D}_w \in \mathbb{R}^{k \times w}$ hidden states from DC-CNN in decoder. We apply Batch normalization and LeakyReLU as an activation function to compute final representations (4). Then, the reconstructed series $\tilde{\boldsymbol{y}}_t$ can be obtained using time step-wise feedforward $g(\cdot)$:

$$\tilde{\boldsymbol{y}}_t = g(\boldsymbol{h}_t) \tag{5}$$

where feedforward function $g(\cdot)$ is performed at each time step separately, but shared weights.

We supervise the entire model with Mean Absolute Error between the true time series and the reconstructed series:

$$\mathcal{L}_1 = \frac{1}{|w|}\sum_{t=1}^{w}\sum_{i=1}^{n}\left(y_t^i - \tilde{y}_t^i\right) \tag{6}$$

where y_t^i denotes i-th time series at time step t.

TABLE I. STATISICS OF THE REAL-WORLD DATASETS

Dataset	# of Train	# of Test	Anomaly (%)
ECG			
(A) chfdb_chf01_275	1,833	1,841	14.61
(B) chfdb_chf13_45590	2,439	1,287	12.35
(C) chfdbchf15	10,863	3,348	4.45
(D) ltstdb_20221_43	2,610	1,121	11.51
(E) ltstdb_20321_240	2,011	1,447	9.61
(F) mitdb_100_180	2,943	2,225	8.38
Gesture	8,451	2,800	24.63
Power demand	1,513	1,596	11.44

E. Anomaly Scoring

Similar to the previous studies, we define the reconstruction error $\boldsymbol{a}_w$ as $\boldsymbol{y}_{\mathrm{t}} - \tilde{\boldsymbol{y}}_t$. In the validation set, we fit the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for all $\boldsymbol{a}_w$ to detect anomalies in the test set. We estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using empirical mean and variance for the reconstruction errors in the validation dataset. Based on the estimation, we define anomaly score at time t in test set as follows:

$$\boldsymbol{s}_t = (\boldsymbol{a}_t - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{a}_t - \boldsymbol{\mu}) \tag{7}$$

We predict the time step t as an anomaly if the anomaly score $\boldsymbol{s}_t$ is greater than a threshold. To dampen an additional hyperparameter, we compute the area under the ROC and Precision-Recall curves, which are widely used measures for anomaly detection.

IV. EXPERIMENTS

A. Datasets

To evaluate TCAE in real-world datasets with labeled anomalies, we obtained publicly available eight benchmark datasets (Table 1) from [9]. Since test sets that contain a small number of anomalies are provided separately, we do not need to divide the test sets. On the other hand, we divide the training datasets, which consist of only normal datapoints, into 70% for train and 30% for validation sets. Also, we scale all datasets using standard normalization.

- Electrocardiograms (ECG): ECG dataset is a collection of six bivariate time series for heart beating records from six people.
- Gesture: Gesture dataset has bivariate time series for the X and Y coordinates of the actor's hand while manipulating a replica gun in the video images. Anomalies are marked if the actor misses the gun in the repetitive movements.
- Power Demand: Power demand dataset contains univariate time series for power consumption, recorded by Dutch Research Institute.

B. Baselines

We compare our model against five state-of-the-art baselines for time series anomaly detection: RAE [6], RAE-Ensemble [10], RRN [9], and RAMED [11]. RAE is comprised of an autoencoder with the recurrent model. RAE-Ensemble exploits an ensemble method for encoders and decoders in which have sparse connections. RRN utilizes self-attention to enhance the encoder-decoder framework. Also, RAMED builds multiple decoders to decode time series in different resolutions.

C. Evaluation Metrics

To avoid introducing specific thresholds in measuring the performance, we use three standard metrics for anomaly detection: AUROC (area under the ROC curve), AUPRC (area under the precision-recall curve), and the highest F1 score. The highest F1 score is chosen among the scores computed using 500 thresholds which are evenly spaced in the interval from zero to the maximum score.

D. Implementation Details

We set the window size w to 256 for all datasets and the size of the kernel to 7 for each convolutional layer. The number of kernels varies depending on the datasets; we conduct grid search from $\{8, 16, 32\}$. Similarly, the size of the latent vector is chosen from $\{4, 8, 16\}$ and the number of stacked dilated causal convolution (M) is chosen from $\{2, 4, 6\}$. We train the whole model using Adam optimizer with a learning rate of 1e-3 and early stop the training with the patience of 10.

E. Anomaly Detection Performance

Table 2 summarizes the experimental results of AUROC, AUPRC, and the best F1 (F1) for all baselines and TCAE on eight datasets. We report published results in [11] since we have the same experimental setting, i.e., train and test sets are provided separately for all datasets. As shown in the table, the proposed TCAE model consistently outperforms all baselines, achieving the best performance on 20 cases of a total of 21 test cases. It shows that the proposed model clearly advances the state-of-the-art time series anomaly detection methods.

In particular, TCAE shows significant performance improvements (about 10% on average) compared to the previous best RNN-based reconstruction methods. This demonstrates that temporal convolution is beneficial to reconstruct the given time series even though it does not have the architectural advantages of being recurrent. In effect, the most of anomaly detection models that impose recurrent neural models need particular technique such as sparse connections, time-reverse decoding, or ensemble approaches to make efficient modeling for temporal dependencies. This indicates that our model is able to learn better representations than other baselines.

Furthermore, we highlight that our model is not sensitive for training and test window, while RAMED [11], the previous best model, uses a relatively short window (64 for ECG and Gesture, and 512 for Power demand datasets).

TABLE II. ANOMALY DETECTION RESULTS IN TERMS OF AUROC, AUPRC, AND F1 SCORES

Metrics	Models	ECG						Gesture	Power demand
		A	B	C	D	E	F		
AUROC	RAE	0.673	0.750	0.829	0.545	0.797	0.472	0.760	0.612
	RRN	0.639	0.762	0.741	0.632	0.810	0.453	0.753	0.661
	RAE-Ensemble	0.688	0.779	0.857	0.640	0.804	0.523	0.781	0.659
	RAMED	0.713	0.855	**0.874**	0.647	0.883	0.640	0.784	0.679
	TCAE	**0.723**	**0.873**	0.858	**0.709**	**0.904**	**0.725**	**0.820**	**0.704**
AUPRC	RAE	0.550	0.425	0.500	0.144	0.213	0.090	0.498	0.135
	RRN	0.856	0.565	0.414	0.165	0.321	0.083	0.487	0.145
	RAE-Ensemble	0.555	0.477	0.526	0.203	0.280	0.095	0.529	0.140
	RAMED	0.580	0.701	0.549	0.220	0.378	0.125	0.533	0.163
	TCAE	**0.603**	**0.746**	**0.577**	**0.267**	**0.445**	**0.150**	**0.589**	**0.255**
F1	RAE	0.548	0.474	0.505	0.219	0.389	0.158	0.530	0.280
	RRN	0.544	0.550	0.454	0.262	0.455	0.156	0.524	0.293
	RAE-Ensemble	0.548	0.502	0.533	0.274	0.391	0.160	0.551	0.268
	RAMED	0.576	0.687	0.554	0.347	0.486	0.209	0.563	0.293
	TCAE	**0.602**	**0.758**	**0.582**	**0.372**	**0.508**	**0.258**	**0.568**	**0.351**

TABLE III. VARYING WINDOW SIZE w ON GESTURE DATASET

Window size w	AUROC	AUPRC	F1
$w = 64$	0.747	0.515	0.551
$w = 128$	0.785	0.539	0.567
$w = 256$	**0.788**	**0.589**	**0.568**
$w = 512$	0.782	0.560	0.566

TABLE IV. VARYING THE NUMBER OF STACKS M ON GESTURE DATASET

# of stacks M	AUROC	AUPRC	F1
$M = 2$	0.702	0.519	0.448
$M = 4$	0.754	0.529	0.536
$M = 6$	**0.788**	**0.589**	**0.568**

F. Hyperparameter Sensitivity Study

To further investigate our model, we conducted additional experiments on Gesture dataset to study hyperparameter sensitivity. Since TCAE mainly imposes 1D CNNs inside the encoder and decoder, we consider the relationships between the input window size and the number of stacks in dilated CNN. Unlike RNNs, in which the performance is highly dependent on the length of the input window, 1D CNNs are not affected by the size since CNNs mainly learn the kernels by striding. However, the receptive field on the top of the stacked CNNs matters to the performance since it represents how far the models look back.

Table 3 shows the performance with different input window size w while fixing M=6. We vary the window size w=64, 128, 256, and 512. We observe that small window 64 slightly degrade the anomaly detection performance, indicating that small input window size may not be helpful due to a large number of paddings for dilation.

On the contrary, Table 4 reports the anomaly detection results using different M=2, 4, 6 with w=256. We clearly confirm that a small number of stacks (M=2) degrades the performance results, owing to insufficient lookback for the available window size. From the results, we recommend using at least four stacks for a window size of 256.

V. CONCLUSION

In this paper, we propose an effective method named Temporal Convolutional Autoencoder, which imposes convolutional networks as encoders and decoders. Our model leverages the entire latent vectors at all time steps to reconstruct the given time series while maintaining causality by dilated causal convolutional operations. Our model utilizes latent vectors as the information for the previous steps instead of the outputs of decoders to eliminate autoregressive decoding. Experiments on real-world datasets demonstrate that our model outperforms baselines with a large margin. Future work includes reconstructing high-dimensional time series for anomaly detection.

REFERENCES

[1] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021.

[2] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, 2017.

[3] Z. Ding, B. Yang, Y. Chi, and L. Guo, "Enabling smart transportation systems: A parallel spatio-temporal database approach," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1377–1391, 2015.

[4] C.-C. Chia and Z. Syed, "Scalable noise mining in long-term electrocardio graphic time-series to predict death following heart attacks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 125–134, 2014.

[5] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[6] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016

[7] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun, "Learning discriminative reconstructions for unsupervised outlier removal," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1511–1519, 2015.

[8] T. Tagawa, Y. Tadokoro, and T. Yairi, "Structured denoising autoencoder for fault detection and analysis," in *Asian Conference on Machine Learning*, pp. 96–111, PMLR, 2015.

[9] Y.-H. Yoo, U.-H. Kim, and J.-H. Kim, "Recurrent reconstructive network for sequential anomaly detection," *IEEE transactions on cybernetics*, vol 51, pp. 1704-1715, 2019.

[10] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles.," in *IJCAI*, pp. 2725–2732, 2019.

[11] L. Shen, Z. Yu, Q. Ma, and J. T. Kwok, "Time series anomaly detection with multiresolution ensemble decoding," in *Proceedings of the AAAI Con-ference on Artificial Intelligence*, vol. 35, pp. 9567–9575, 2021.

[12] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier analysis*, pp. 1–34, Springer, 2017.

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[14] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, pp. 473– 479, 1997.

[15] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.047*3, 2014.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[18] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[19] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[20] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN),* pp. 1578–1585, IEEE, 2017.

[21] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[22] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[23] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *arXiv preprint arXiv:1901.10738*, 2019.