A Comparison of Classical and Deep Learning based Anomaly Detection Methods in Multivariate Time Series

Anay Pattanaik and Yuchen Liang

July 22, 2022

1 Introduction

Time series is one of the fundamental statistical models for sequential observations, and it has found many applications in robotics and finance. Traditional time series techniques include estimation and prediction, but in some applications, detection, especially anomaly detection, is the primary task. In stock markets, for example, an accurate anomaly detection system would enhance financial model building and even business decision-making processes [1]. In the era of big data, the dimensions of observations at each time step is usually large, so it is rewarding to consider anomaly detection in Multivariate Time Series (MVTS).

There are three kinds of anomalies: point anomalies (where a single point is itself an anomaly), contextual anomalies (where a point is anomalous in a given context), and collective anomalies (where a set of points are anomalous while each a single point is not). In MVTS, we are primarily interested in the latter two definitions for anomalies. Anomaly detection is related but fundamentally different from sequential change (or change-point)

detection. In change detection problems, the goal is to detect a change in distribution as quickly as possible, subject to false alarm constraint (See [2] for a survey). The post-change distribution is usually assumed to be persistently different from that of the pre-change. In comparison, anomalies, though possibly contextual or collective, are by definition not persistent in nature. Moreover, the goal of anomaly detection is to accurately identify each anomalous point in the data stream.

Anomaly detection methods can be divided into three categories: supervised, semi-supervised, and unsupervised. In this paper, we are interested in semi-supervised methods, where we chose the training data only under normal operations. We then set a threshold as a function of either the likelihood of fitted model or use reconstruction error (for generative model). The anomaly is detected if any data point exceeds this threshold. We will compare classical and deep learning based anomaly detection methods in MVTS. Classical methods draw techniques from traditional model fitting (e.g., Gaussian and ARIMA) and traditional machine learning (e.g., Isolation Forest). Deep learning methods include those based on the encoder-decoder model, namely Fully connected Auto Enoder (FAE) [3] and Long-short term memory Encoder-decoder (LSTM-ED) [4]. Finally, we compare their performance with classical methods.

We refer interested readers to [5, 6] for a more complete survey for anomaly detection.

2 Problem Statement

Suppose that we are given observations under normal operations $\boldsymbol{X}_{train} \in \mathbb{R}^{n_{train},d}$, where n_{train} is the number of training data, and d is the dimension. The goal is, given the test data $\boldsymbol{X}_{test} \in \mathbb{R}^{n_{test},d}$, to classify each test data as anomalous (1) or not (0). Note that we are not

interested to identify which particular dimensions are anomalous.

3 Performance Metric and Dataset

The performance metrics used in this paper is: accuracy, precision, recall, and F1 scores. They are defined as:

$$Accy(M) := \frac{\sum_{i=1}^{n_{test}} \mathbb{1}\left\{y^{(i)} = \hat{y}^{(i)}\right\}}{n_{test}}, \quad Prec(M) := \frac{\sum_{i=1}^{n_{test}} \mathbb{1}\left\{y^{(i)} = 1, \hat{y}^{(i)} = 1\right\}}{\sum_{i=1}^{n_{test}} \mathbb{1}\left\{\hat{y}^{(i)} = 1\right\}}$$
$$Rec(M) := \frac{\sum_{i=1}^{n_{test}} \mathbb{1}\left\{y^{(i)} = 1, \hat{y}^{(i)} = 1\right\}}{\sum_{i=1}^{n_{test}} \mathbb{1}\left\{y^{(i)} = 1\right\}}, \quad F1(M) := \frac{2 \times Prec(M) \times Rec(M)}{Prec(M) + Rec(M)}$$

where M is the model to be evaluated, $y^{(i)}$ is the ground truth for $\boldsymbol{X}_{test}^{(i)}$, and $\hat{y}^{(i)}$ is the predicted label using model M.

We will use two real-world datasets for evaluation. The first one is the Skoltech Anomaly Benchmark (SKAB) which is data from a water circulation system [7]. The dimension of data is 8, where each corresponds to a particular sensor in the system (e.g., voltage, pressure, temperature). The goal is to identify regions where the circulation system is working abnormally. A sample test data is plotted in Fig. 1.

The second dataset is the Server Machine Dataset (SMD) which is data from 28 server machines [8]. The dimension of data is 38, and each dimension describes a particular server state (e.g., CPU load, network usage, and memory usage).

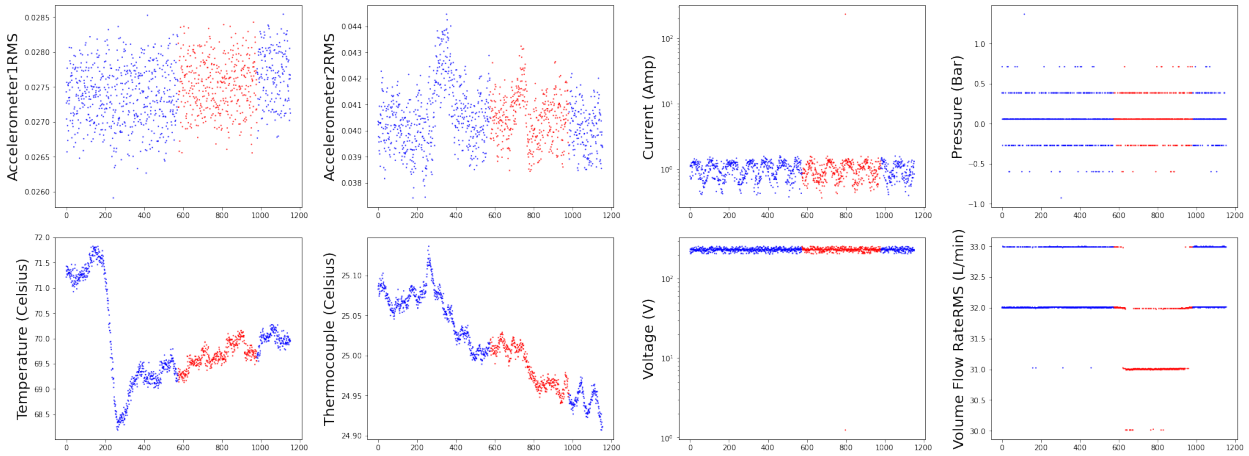


Figure 1: A sample SKAB test data. The blue dots are data under normal operations, and the red dots are abnormal points.

4 Anomaly Detection Methods

4.1 General Framework

We first introduce the general anomaly detection framework for MVTS in Fig. 2, adapted from [6]. The key is to obtain some scoring function and a threshold from the training samples, where the former will behave as the test statistic and the latter, the test threshold. Different anomaly detection methods below use different ways to construct the scoring function and the threshold. A baseline detection method will first be introduced in 4.2. Then, two classical methods, the ARIMA based method and the Isolation Forest method will be introduced in 4.3. The ARIMA based method tries to capture the time dependencies, and the Isolation Forest method is a well-known anomaly detection method using traditional machine learning techniques. Finally, several deep learning based methods will be presented in 4.4.

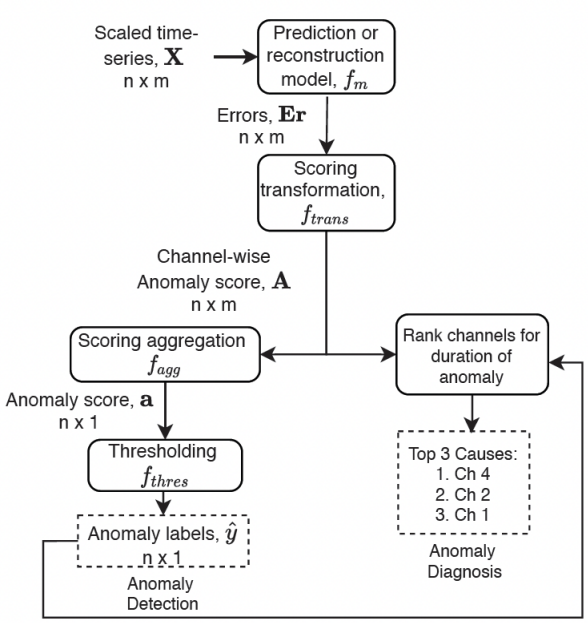


Figure 2: General Framework of Anomaly Detection in MVTS. The figure is adapted from [6]. The difference is that in this paper we are not interested in Anomaly Diagnosis, and we simply output anomaly labels $\hat{y}$.

4.2 Baseline method

The idea for the baseline method is simple: Given the training data (under normal operations), a Gaussian distribution is fit on the training data. For practical computation reasons, a univariate Gaussian is fit on each dimension, and independence is assumed across all dimensions. During testing, a test point $x^{(i)}$ is labelled as anomalous if

$$\sum_{j=1}^{d} \widehat{\Phi}_j(x_j^{(i)}) < \alpha \quad \text{or} \quad \sum_{j=1}^{d} \widehat{\Phi}_j(x_j^{(i)}) > 1 - \alpha/2$$

and the threshold $\alpha = 0.05$, where $\widehat{\Phi}_j$ is the CDF for the fitted univariate Gaussian on dimension j.

4.3 Classical methods

4.3.1 ARIMA based

To model time-domain structure, we incorporate the ARIMA model in the anomaly detection framework. First, the PCA features of the training data are used in order to remove feature correlation. Then, an ARIMA model is fit on each dimension of the uncorrelated training data. Finally, the test data is compared to the forecasted values from the fitted model. If the sum of residuals (across all dimensions) are too large (where the threshold is set to be the sum of variances), an anomaly is declared.

4.3.2 Isolation Forest

Isolation Forest is a traditional anomaly detection method for non-sequential data [9]. It uses an ensemble of a particular type of decision trees, called Isolation Trees, and cleverly observes that anomalous points on average require fewer decisions to be made before they are isolated from the rest of the data in a binary decision tree. Thus, given a test point, the authors propose the tree path length as the scoring function, and the threshold is related to the mean height of the trees. Although Isolation Forest is an unsupervised anomaly detection algorithm that deals with non-sequential data, we use it in our experiment where all training points are normal and are sequenced in time. In our experiment, we use default parameters from the Scikit-Learn package, many of which are recommended in the original paper.

4.4 Deep Learning based methods

In this subsection, we will present two deep learning techniques for anomaly detection and compare their performance with classical methods.

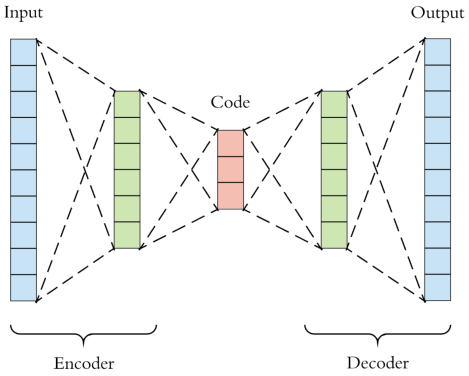
4.4.1 Fully-connected Auto Encoder (FAE) [3]

Autoencoders were originally developed for dimensionality reduction, that is, to compress the data. An autoencoder consists of an encoder that maps input to the latent space (code) and a decoder reconstructs the input from the latent space as shown in Fig. 3a. usually both the encoder and decoders are neural networks and the loss function is the reconstruction error. This loss function is minimized in an end-to-end fashion. We can state this formally as follows: $\phi : \mathcal{X} \to \mathcal{Z}$ is the encoder that maps the input space $\mathcal{X}$ to the latent space $\mathcal{Z}$ and $\psi : \mathcal{Z} \to \mathcal{X}$ is the decoder that reconstructs the input/observation given the latent representation. The loss function is usually the reconstruction error, that is, $L(\phi, \psi) = ||x - (\psi \circ \phi)x||^2$. In fully connected autoencoder architecture, the neural networks are feedforward in nature.

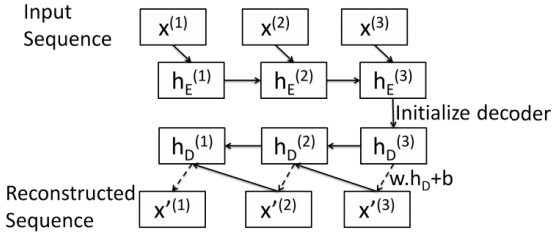
For the purpose of anomaly detection, the reconstruction error of the input can be used as an threshold for detection of the anomaly. Notice that only non-anomalous data is used for training so that anomalous input will result in large reconstruction error. This threshold is selected to maximize the F1 score on validation set.

4.4.2 LSTM Encoder-Decoder (LSTM-ED) [4]

The LSTM encoder-decoder consists of LSTM neural network instead of fully conned neural networks as encoder and decoder. It is trained to reconstruct the non-anomalous time series



(a) Fully connected Autoencoder[10]



(b) LSTM Autoencoder[4]

Figure 3: Auto encoder architectures

as shown in Fig. 3b. Here $h^{(i)}(x)$ represents the latent state and $x^{(i)}$ is the time series. The model is trained to minimize the loss function given $\sum_{x \in \mathcal{S}_N} \sum_{i=1}^{L} ||x^{(i)} - x'^{(i)}||^2$ where $\mathcal{S}_N$ represents the non-anomalous data points.

Next we discuss the thresholding technique to detect the anomaly. The reconstruction error at time t_i is given as $e^{(i)} = |x^{(i)} - x'^{(i)}|$, these error vectors are then used to fit a multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ using the maximum likelihood method. Then the likelihood from this fitted multivariate normal is used for anomaly score $a^{(i)} = (e^{(i)} - \mu)^T \Sigma^{-1} (e^{(i)} - \mu)$. The threshold for detecting anomaly is chosen by maximizing the F1 score on a validation set.

5 Numerical Performances and Discussion

In this section, we present the performance of each method described in Sec. 4 with experiments on SKAB and SMD datasets. The results are recorded below in Table 1 and Table 2. We can observe that the performance of deep learning algorithms (FAE and LSTM-ED) is consistently better than classical methods. The difference in performance is even more

apparent in SMD dataset. A plausible explanation could be the fact that SMD dataset is a higher dimensional and larger sized dataset. SMD dataset is 38 dimensional with 708,405 training instances as compared to 8 dimensional SKAB dataset with about 15,000 training instances. So, it is not very surprising that the deep learning methods perform better with more challenging datasets. Among all the four approaches, the fully connected autoencoder (FAE) performed the best as shown in Table 1 and 2.

Model	**Accy**	**Prec**	**Rec**	**F1**
Baseline	0.353	0.353	0.995	0.521
ARIMA based	0.356	0.354	0.994	0.522
Isolation Forest	0.374	0.358	0.971	0.523
Fully connected Autoencoder	0.430	0.412	0.888	**0.563**
LSTM-ED	0.422	0.409	0.885	0.559

Table 1: Performance on SKAB Dataset.

Model	**Accy**	**Prec**	**Rec**	**F1**
Baseline	0.042	0.042	1.0	0.080
ARIMA based	0.245	0.005	0.98	0.011
Isolation Forest	0.690	0.080	0.616	0.142
Fully connected Autoencoder	0.897	0.815	0.310	**0.450**
LSTM-ED	0.880	0.875	0.270	0.413

Table 2: Performance on SMD Dataset.

6 Conclusion

Thus, we compared classical and deep learning techniques for anomaly detection in two different datasets. The deep learning methods perform much better than classical techniques with higher dimensional (and larger) datasets, and their performance is at par with classical methods in relatively easy datasets.

References

[1] M. Ahmed, N. Choudhury, and S. Uddin, "Anomaly detection on big data in financial markets," in *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 998–1001, 2017.

[2] L. Xie, S. Zou, Y. Xie, and V. Veeravalli, "Sequential (quickest) change detection: Classical results and new directions," *IEEE Journal on Selected Areas in Information Theory*, vol. PP, pp. 1–1, 04 2021.

[3] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[4] 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.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, jul 2009.

[6] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2021.

[7] I. D. Katser and V. O. Kozitsin, "Skoltech anomaly benchmark (skab)," 2020.

[8] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data*

Mining, KDD '19, (New York, NY, USA), p. 2828–2837, Association for Computing Machinery, 2019.

[9] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, mar 2012.

[10] A. F. Gad, "Image compression using autoencoders in keras," 2020.