Robust Unsupervised Anomaly Detection with Variational Autoencoder in Multivariate Time Series Data

著者	Yokkampon Umaporn
その他のタイトル	多変量時系列データの変分オートエンコーダによるロバストな教示なし異常検知
学位授与年度	令和4年度
学位授与番号	17104甲情工第370号
URL	http://doi.org/10.18997/00009009

Robust Unsupervised Anomaly Detection with Variational Autoencoder in Multivariate Time Series Data

(多変量時系列データの変分オートエンコーダによるロバストな教示なし異常検知)

UMAPORN YOKKAMPON

Table of contents

Page

Chapter 1
Introduction

1.1 Background

In data mining, a time series [1] is a sequence of data points collected over time [2]. Such a sequence forms the basis of methods for tracking changes over time. Time series data can track changes in milliseconds, days, months, or even years. Time series data plays an important role in virtually all areas of science, engineering, commerce, and industry. Since data points in time series are collected at intervals, there is a relationship between successive observations, proportional or not, which distinguishes time series data from other kinds of data. There are two types of time series data. One is a univariate time series based on a single time dependent variable (or dimension). The other is a multivariate time series based on two or more time dependent, interrelated variables (or dimensions). One of these important fields of time-series is anomaly detection. Time-series anomaly detection has very important significance and has become a necessary part of the modern manufacturing industry and information services because undetected anomalies may cause serious damage.

Currently, there have been many studies on anomaly detection of time series data [3-6]. Anomaly detection is the identification of unexpected data points, i.e., events or items that differ significantly from what is expected. Three types of time series anomalies have been identified, namely, point anomalies, contextual anomalies, and collective anomalies [7]. Point anomalies are points that exist far outside the range of the entire data set, which can occur in any type of data. Contextual anomalies are values that deviate significantly from most of the data points in the same context, which can only occur in relative data. Collective anomalies occur when a subset of the data points deviates substantially from the entire dataset. Anomaly detection is critical in many real world applications, such as the analysis of potentially fraudulent transactions, sensor network faults, medical diagnosis errors, abnormal equipment behavior, etc. For example, detecting bank transaction fraud could save 32 billion dollars

worldwide by 2020 [8]. Gupta et al. [9] studied abnormal changes in GDP components over time, and Keogh et al. [10] checked whether an electrocardiogram had abnormal fluctuations. Therefore, the industry needs to be able to detect anomalies in its system.

In recent years, due to the development of industry and the Internet of Things [11-15], multivariate time-series anomaly detection technology has made great progress [16-19]. We can obtain more reliable time-series data from the devices by configuring a multisensor system. However, processing these data from sensors is a major problem. First, the data collected by different sensors may have different attributes, frequencies, and dependencies. Therefore, preprocessing these data is a very time-consuming task and may require some domain knowledge. Jin et al. [20] proposed an innovative learning framework for multivariate air pollutant concentration prediction. This method, which separated the features and trends by decomposing the original data into high-frequency parts and low-frequency parts to learn them respectively in a multi-channel module, provided a great idea for us to acquire the features of multivariate time series. In addition to the problems mentioned above, there are still some unavoidable problems; for example, it is difficult to set an accurate boundary for normal and abnormal data, or the data collected by different sensors may contain noise due to other factors. These data with serious noise may look similar to anomalies [19], lead to false alarms [21], and affect the performance of algorithms [7]. The fact that the amount of normal data is much larger than the amount of abnormal data is another problem, and the problem of extremely unbalanced data has become another major trouble spot in time-series anomaly detection [22].

The occurrence of anomalies in multivariate time series data typically involves multiple features. Sequential analysis of individual features cannot accurately locate all the anomalies because several variables have to be examined simultaneously when analyzing data segments. Moreover, encoding the inter-correlations between different pairs of time series also needs to be considered. Clearly, detecting anomalous parts of multivariate time series is a challenging problem.

To establish an automatic detection system of anomalies in time-series, many researchers have proposed many effective models and methods to deal with time-series data. Many scholars have been studying time-series modeling including ARIMA [23], SVM [24], and CNN [19]. Since the data used for anomaly detection usually have no clear labels and the amount of abnormal data is very small, many unsupervised discriminative approaches are used for anomaly detection, including OCSVM [25], iForest [26], and LSTM-ED [27]. Although these unsupervised methods have made some progress in the field of time-series anomaly detection, many models still cannot detect anomalies effectively.

Over the last decade, there has been an increased enthusiasm around deep neural networks (DNNs) [28] which aim to learn deep latent representations of the multivariate time series to infer a model of variability used for anomaly grading in unseen data. As a result of the good performance demonstrated by DNNs in multiple fields [29, 30, 31], in recent years, there has been a boom in DNN-based methods for multivariate time series anomaly detection (Table 1.1).

Machine learning techniques are increasingly being adopted to detect anomalies because they can capture different characteristics of time series and detect anomalies effectively. Various anomaly detection methods for multivariate time series data have been developed. Especially noteworthy are methods based on dimension reduction. These methods aim to reduce the dimension of the space defined by a data set while retaining the important features of the original data. Dimension reduction methods differ according to their handling of feature selection and feature extraction; these methods may be linear or non-linear. The Autoencoder is particularly important in this area. This method attempts to compress and thus map input data to reduced dimensional space and then use an encoding-decoding process to reconstruct the input data set. A newer dimensionality reduction method is Variational Autoencoder (VAE), which evolved from Autoencoder. VAE is a type of neural network that can learn to compress data in a completely unsupervised way. This method outperforms Autoencoder by imposing a probability distribution on the latent space, with a given mean and variance, and using a sample

from this distribution to reconstruct the data. Despite considerable progress, VAE based anomaly detection for imbalanced data has not received much attention.

Therefore, our goal is to study and develop specific approaches to detect anomalies in multivariate time series data, which takes account of the correlation between the series. Multivariate time series data usually contains noise and masks the true anomalies. Also, we need to consider the characteristics of the data itself and consider the threshold because, when facing the imbalance issue of normal and abnormal samples, the existing threshold setting strategy is insensitive to imbalanced datasets. The main contributions of our proposed framework are as follows.

- The novel MSCVAE framework is designed to detect anomalies in multivariate time series data. MSCVAE constructs multi-scale attribute matrices to characterize multiple levels of the system states across different time steps and then uses a convolutional variational autoencoder to extract the characteristics of the time series input. Specifically, we use an attention-based Convolutional Long-Short Term Memory (ConvLSTM) network to capture the temporal patterns and also to reconstruct the attribute matrices. Moreover, the weighted mechanism is introduced into the last layer of decoding, which helps to give different weights to every data point of sequence in each sliding window.
- We propose a novel threshold setting strategy based on a confusion matrix to optimize threshold selection of anomaly detection, which will help to improve the model robustness under conditions of imbalance between normal and abnormal data in multivariate time series.
- Experiments have been conducted on four datasets in order to verify the effectiveness of the proposed framework and the new threshold setting strategy. The results demonstrate that our method is superior to competing models in terms of anomaly detection performance and robustness under different ratios of imbalanced datasets.

Table 1.1 Deep learning-based methods for anomaly detection in multivariate time series from 2018 to 2021

Methods	Description	Datasets
DAGMM [31]	Deep Autoencoding Gaussian Mixture Model	MSL, SMAP, SMD, SWaT, WADI
AE [32]	Autoencoder	MSL, SMAP, SMD, SWaT, WADI
Donut [33]	Variational Autoencoder	Private
USAD [34]	Adversely trained Autoencoders	MSL, SMAP, SMD, SWaT, WADI
Bagel [35]	Conditional Variational Autoencoder	Private
OmniAnomaly [36]	Gated Recurrent Unit and Variational Autoencoder	MSL, SMAP, SMD, SWaT, WADI
MAD-GAN [37]	Generative Adversarial Networks	SWaT, WADI, KDDCUP99
LSTM-VAE [38]	LSTM-Variational Autoencoder	MSL, SMAP, SMD, SWaT, WADI
DeepAnT [5]	Convolutional neural network	Yahoo Webscope
MSCRED [39]	Multi-Scale Convolutional Recurrent Encoder-Decoder	Power Plant, Synthetic
MTS-DCGAN [21]	Deep Convolutional Generative Adversarial Network	Genesis, Satellite, Shuttle, Gamma
FuseAD [40]	ARIMA and Convolutional neural network	Yahoo Webscope, NAB
RADM [41]	Hierarchical Temporal Memory and Bayesian Network	NAB
MTAD-TF [42]	Convolutional and Graph Attention Network	MSL, SMAP, SMD

1.2 Problem statement

Since the main problem is that industries require more than 99% anomaly detection accuracy, we need to consider several reasons to achieve this problem. First, there is a close time dependence between multidimensional time-series data, and general density-based methods and clustering models cannot capture the dependence between series. Second, multivariate time-series from the real world containing relatively severe noise may reduce the generalization ability of the detection model. Finally, the problem of data imbalance will cause the model to be unable to fully obtain the relationship between normal data and abnormal data, which will lead to a poor detection effect. Thus, all problems will be considered and solved by the proposed framework in this thesis.

1.3 Research purpose

The general objective of this research is to study and develop an unsupervised anomaly detection algorithm for multivariate time series data. The specific objectives are established as follows:

1) To create a new framework Multi Scale Convolutional Variational Autoencoder (MSCVAE) for detecting anomalies in multivariate time series.

2) To build the new Error Rate (ERR) based threshold setting strategy of anomaly detection, which will help to improve the model robustness under conditions of imbalance between normal and abnormal data in multivariate time series.

3) To verify the effectiveness of the proposed framework and the new ERR based threshold setting strategy on four benchmark datasets and compare the performance with other algorithms.

1.4 Overview of the thesis

The thesis organization outline consists of five chapters that cover the background history, research objectives, fundamentals of the time series, autoencoder and variational autoencoder, experiment, and discussion of the results. Finally, the conclusions and suggestions for further work are explained in detail and related information. The particular explanation of each chapter will be explained as follows:

Chapter 1 in this introduction chapter explains the background and problem of anomaly detection in multivariate time series data. The objective and contribution are to develop an anomaly detection architecture system that combines the deep learning method to detect anomalies in many practical settings. Finally, this chapter clarifies the particular objective.

Chapter 2 presents the fundamental basis of the time series and its characteristics. Then the problem of anomaly detection in time series is in detail, followed by the concepts of the artificial neural networks, autoencoders, variational autoencoders, and evaluation metrics, and cover all methods used in this thesis.

Chapter 3 outlines the practical methods used for anomaly detection in multivariate time series data and their implementation in the proposed algorithms. The chapter introduces the problem we aim to study and then shows how to generate attribute matrices, which is the pre-processing process. Next, we elaborate on the proposed framework in detail. Furthermore, we introduce in detail the new threshold setting strategy to optimize anomaly detection performance under an imbalance of normal and abnormal data.

Chapter 4 presents the results of the thesis. The chapter begins with an explanation of the four standard datasets used in this thesis. Afterward, the experimental setup and anomaly detection results are presented. This is followed by results from the ablation study, robustness evaluation, and threshold setting strategy comparison.

Finally, chapter 5 conclusion summarizes the thesis with three parts of the proposed framework. Firstly, in pre-processing data, we calculate an attribute matrices based inner-product for each time step, which contains the relationship between its own information and the information of a sub-sequence. That is why we can amplify features and reduce noise. Second, attention-based ConvLSTM is applied to select adaptively relevant hidden states (feature maps) across different time steps. That is why we can capture the temporal patterns. Third, a new error rate (ERR) based threshold setting strategy is applied to optimize anomaly detection performance under an imbalance of normal and abnormal data. Finally, the ideas for improvements for future work are presented.

Chapter 2
Background & Theory

In this chapter, related preliminary knowledge is presented and discussed for a better understanding of this thesis. The chapter begins with an introduction to time series, anomaly detection, artificial neural network, fundamental theories such as Autoencoder, Variational Autoencoder, and evaluation metrics, which are used to detect the anomalies in multivariate time series data.

2.1 Time Series

2.1.1 Definition of time series

A time series is defined as a sequence of ordered continuous values representing a numerical variable's evolution over time. It is the measurement of a system evolving in time with numerical attributes: for example, the temperature of a computer server, the value of a company's stock, or the electrical activity of the heart (ECG). Therefore, a time series is any sequence of observations indexed by time. A time series carries a lot of information about the measuring system. This information can be used to ensure the proper functioning of the system.

Time series are used in statistics, signal processing, pattern recognition, finance, weather forecasting, astronomy, communications engineering, and largely in any applied science and engineering domain that involves temporal measurements.

Time series analysis comprises methods for analyzing time-series data to extract meaningful statistics and other data characteristics. The difference between a simple regression task and a time series analysis is that, in the latter case, the model must not only learn the correlation between characteristics but also the correlation with time.

2.1.2 Univariate and Multivariate time series

Researchers have defined two main categories of time series.

Definition 1. (Univariate time series) A univariate time series $X = \{x_t\}_{t \in T}$ is defined as an ordered set of real-valued observations, x_t , where each observation is recorded at a specific $t \in T \subseteq \mathbb{Z}^+$.

Definition 2. (Multivariate time series) A multivariate time series $X = \{x_t\}_{t \in T}$ is defined as an ordered set of k-dimensional vectors, each of which is recorded at specific time $t \in T \subseteq \mathbb{Z}^+$ and consists of k real-valued observations, $x_t = (x_{1t}, ..., x_{kt})$.

A univariate detection method only considers a single time-dependent variable, whereas a multivariate detection method is able to simultaneously work with more than one variable.

Moreover, the detection method can be univariate even if the input data is a multivariate time series because an individual analysis can be performed on each time-dependent variable without considering the dependencies that may exist between the variables.

In contrast, a multivariate technique cannot be used if the input data is a univariate time series.

2.1.3 Autocorrelation

Since a time series is a sequence of values for different timestamps, it could be useful to find the temporal correlation within the same features.

Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of the same feature of a time series (hence the name autocorrelation).

There are several autocorrelation coefficients, corresponding to each panel in the lag plot. For example, r_1 measures the relationship between y_t and y_{t-1}, r_2 measures the relationship between y_t and y_{t-2} and so on.

The value of r_k can be written as:

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})} \tag{2-1}$$

where T is the length of the time series. The autocorrelation coefficients make up the autocorrelation function or ACF.

The autocorrelation plot can also be used to view if the time series has a trend or seasonal behavior. When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. Therefore, the ACF of trended time series tends to have positive values that slowly decrease as the lags increase.

When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags.

When data are both trended and seasonal, these effects are combined.

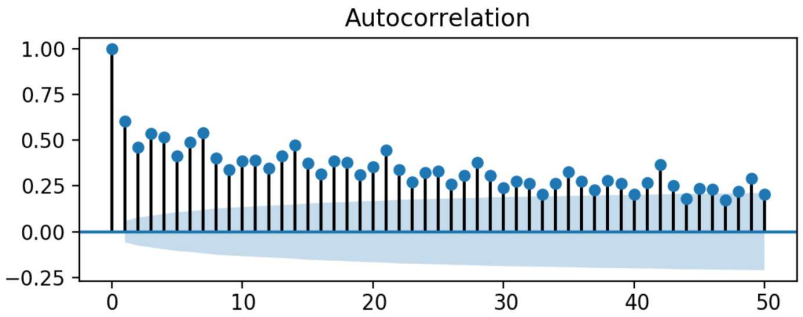


Fig. 2-1 Autocorrelation plot

The autocorrelation plot in Figure 2-1 represents the temporal correlation of the temperature recorded in Barcelona during 2019. It is possible to observe that the correlation between a day and the four preceding days is very strong, this means that the temperature of one day depends on one of the days immediately before. Moreover, the autocorrelation plot has peaked with lags equal to 10 and 30 (one month); thus, for a given day dt has a strong correlation with the temperature of dt−1:t−4 but also with temperature further back in time. We can conclude by saying that the time series considered as an example is stationary with no associated trend because the autocorrelation decreases quickly for small lags but with a likely seasonal behavior.

2.1.4 Decomposition

Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components (trend, seasonality, and cycles), each representing an underlying pattern category.

During the decomposition, the trend and cycle are usually combined into a single trend-cycle component. Hence, a time series can be viewed as a combination of three components: a trend-cycle component, a seasonal component, and a remainder component (containing anything else in the time series).

Often this is done to help improve understanding of the time series, but it can also be used to improve forecast accuracy.

When decomposing a time series, it is sometimes helpful to first transform or adjust the series in order to make the decomposition (and later analysis) as simple as possible.

An additive decomposition is when:

$$y_t = S_t + T_t + R_t, \tag{2-2}$$

where y_t is the data, S_t is the seasonal component, T_t is the trend-cycle component, and R_t is the remainder component, all at period t. Alternatively, a multiplicative decomposition would be written as

$$y_t = S_t \times T_t \times R_t. \tag{2-3}$$

The additive decomposition is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series. When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative decomposition is more appropriate. Multiplicative decompositions are common with economic time series.

Many time series include trends, cycles, and seasonality. When choosing a forecasting method, the first step is to identify patterns in the time series data, and then choose a method that is able to capture those patterns properly.

2.1.5 Trend

Trend is a pattern in data that shows the movement of a series to relatively higher or lower values over a long period of time. In other words, a trend is observed when there is an increasing or decreasing slope in the time series. The trend usually happens for some time, then disappears, and it does not repeat.

In Figure 2-2, the antidiabetic drug sales in Australia show a clear and increasing trend of sales during the years.

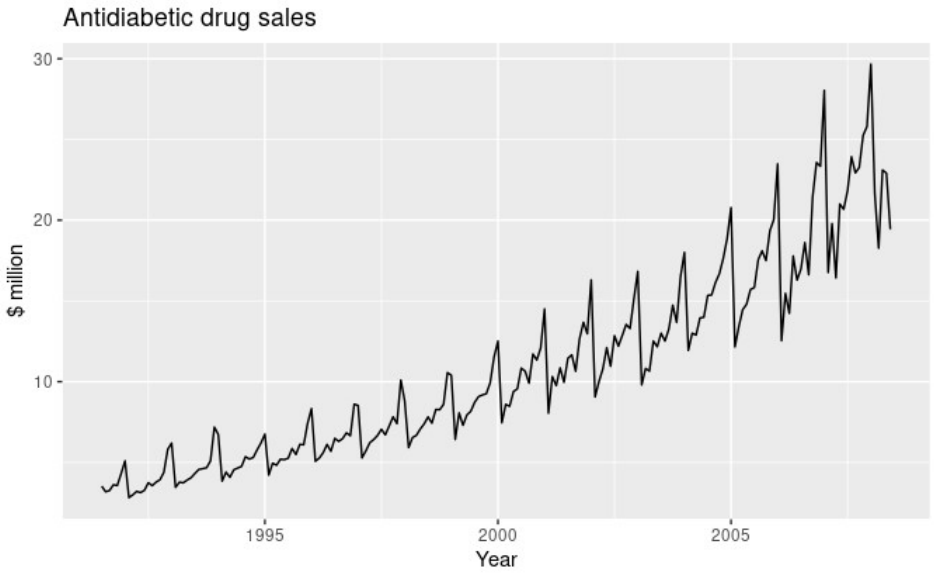


Fig. 2-2 Monthly sales of antidiabetic drugs in Australia [43]

2.1.6 Seasonality

A seasonal pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always a fixed and known period.

For example, the monthly sales of antidiabetic drugs (Figure 2-3) show seasonality which is induced partly by the change in the cost of the drugs at the end of the calendar year.

In this case, it is clear that there is a large jump in sales in January each year. Actually, these are probably sales in late December as customers stockpile before the end of the calendar year, but the sales are not registered with the government until a week or two later. The graph also shows that there was an unusually small number of sales in March 2008 (most other years show an increase between February and March). The small number of sales in June 2008 is probably due to the incomplete counting of sales at the time the data were collected.

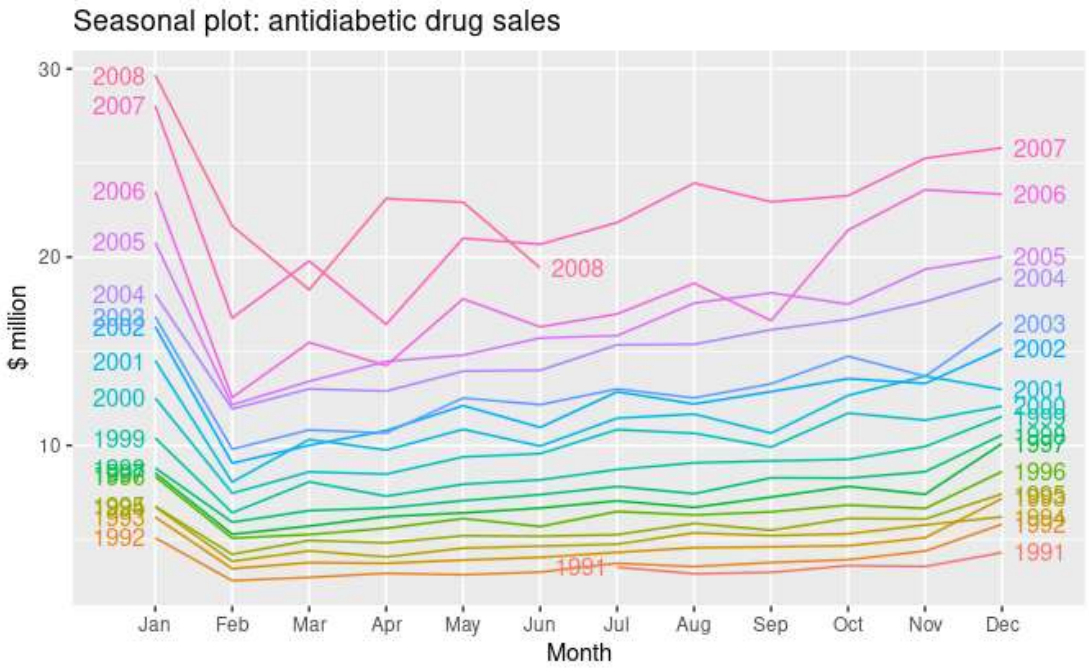


Fig. 2-3 Example of seasonality [43]

2.1.7 Cycles

A cycle occurs when the data exhibit rises and falls that are not of a fixed frequency. These fluctuations are usually due to economic conditions, and are often related to the "business cycle." The duration of these fluctuations is usually at least 2 years.

Cyclic behavior is quite different from seasonal behavior. If the fluctuations are not of a fixed frequency, then they are cyclic; if the frequency is unchanging and associated with some aspect of the calendar, then the pattern is seasonal.

In general, the average length of cycles is longer than the length of a seasonal pattern, and the magnitudes of cycles tend to be more variable than the magnitudes of seasonal patterns.

The monthly housing sales in the USA, in Figure 2-4, show strong seasonality within each year, as well as some strong cyclic behavior with a period of about 6–10 years.

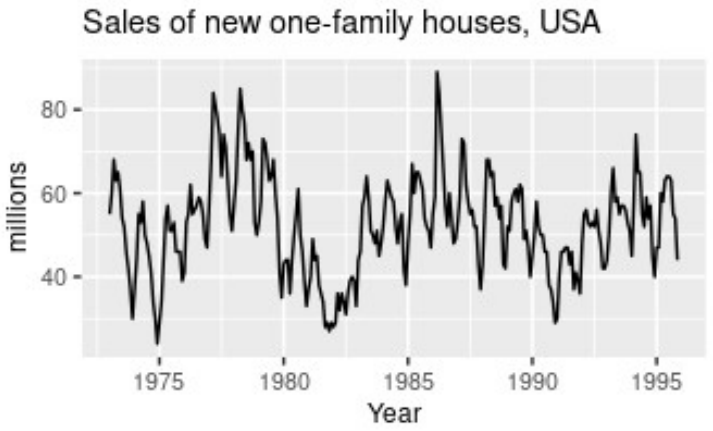


Fig. 2-4 Example of a cycle [43]

2.1.8 Stationarity

Intuitively, a stationary time series is a time series having the same characteristics over every time interval, or in other words, whose properties do not depend on the time at which the series is observed. Formally, we can express it as follow:

Definition 3. X_t is a stationary time series, if $\forall s \in \mathbb{R}$: the distribution of $(x_t,...,x_{t+s})$ is equal.

The above definition implies that a stationarity time series $x_t,...,x_T$ will have the following characteristics:

1. Constant mean, Therefore, no trend exists in the time series.
2. The time series has constant variance.
3. There is a constant autocorrelation over time.
4. The time series has no seasonality, i.e., no periodic fluctuations.

Most of the time series is not stationary, but some methods could help to make the data close to the stationarity.

Differencing One of the most used methods is differencing because it can happen that a time series is not stationary, but the differences between consecutive observations are. Therefore the time series after the transformation is given as $x'_t = x_t - x_{t-1}$.

Is it always better to have stationary time series? Machine learning methods are used when the classical methods fail, and better results are needed. It is impossible to know how to best model unknown nonlinear relationships in time series data, and some methods may result in better performance when working with non-stationary observations or some mixture of stationary and non-stationary views of the problem.

In conclusion, stationary time series are not always preferred, but this is part of the feature engineering/selection when using machine learning methods.

2.2 Anomaly Detection

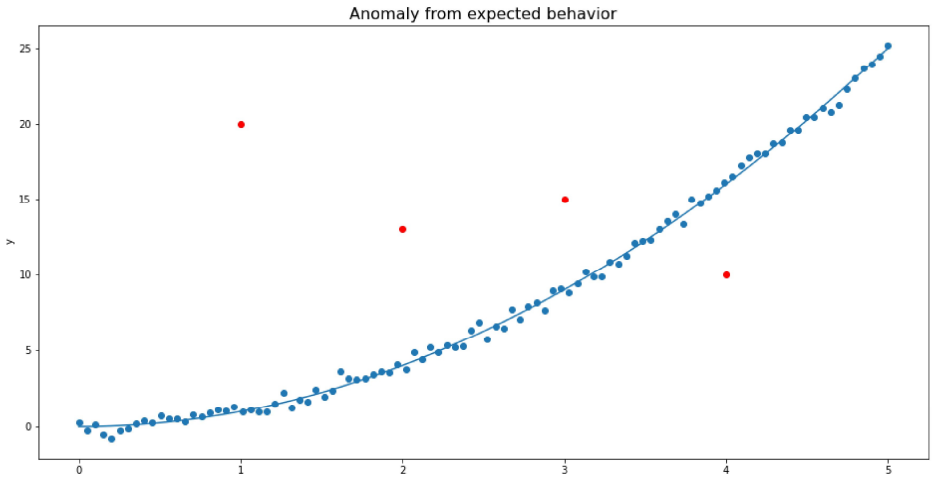


Fig. 2-5 Example of Anomaly from expected behavior

An anomaly can be defined as an unexpected observation with respect to a set of other pre-established observations considered normal. More formally, in a set X containing n observations noted x_i, then $x_p \in A$ will be considered as abnormal if it differs, by its characteristics, from the other observations, i.e., from those contained in the set $A/\{x_p\}$. The definition of the term anomaly is specific to the use case. The most common one in the field of

detection is an observation that is different from the others by its singularity: it could result from a set of rules which are different from the other observations [44]

Anomalies in time series, also called outliers, are points or sequences of points that do not correspond to normal behavior [2]. The concept of normal behavior is difficult to formalize. Therefore, another possible definition for anomalies could be a pattern in data that is not expected compared to what has been seen before [2]. In fact, an implicit assumption is that anomalies are rare events. Anomalies should not be confused with the noise present in the time series. Noise is a phenomenon that, unlike anomalies, has less interest in being analyzed.

However, anomalies may indicate a significant problem in several applications. For example, an anomaly in industrial control systems may indicate a malfunction, financial anomalies may be the result of fraud, or they may indicate diseases in healthcare. As a critical task, many methods have been developed to address it [45, 46].

Anomaly detection refers to the task of identifying an unseen observation $\hat{x}_t, t > T$, based on the fact that it differs significantly from X, thus assuming that X contains only normal points. The amount by which the unseen sample $\hat{x}_t$ and the normal set X differ is measured by an anomaly score, which is then compared to a threshold to obtain an anomaly label.

2.2.1 Types of Anomalies in Time Series

In recent years, due to the increase of the complexity of the reality that we want to model, also the complexity of the anomalies is increased. Therefore, it is necessary to have tools to analyze such data, learn the patterns, and autonomously detect anomalies.

For this reason, Deep anomaly detection (DAD) methods have been shown to detect all three types of anomalies with great success.

Deep learning models can detect anomalies in both univariate and multivariate data, whether the anomaly afflicts a single sensor or more time-dependent variables. In other words, we can find anomalies in one or more features in the case of multivariate time series.

There are mainly three types of anomalies that are studied in the literature, namely point anomalies, sequential anomalies, and contextual anomalies [2]. A brief description of those is given below.

- **Point Anomalies**: If a single point deviates from the considered normal pattern it is referred to as a point anomaly. This is the simplest form of an anomaly. An example of a point anomaly is if a process value suddenly is very low or high. An illustration of this is given in Figure 2-6(left), where the anomaly is marked in red.
- **Sequential Anomalies**: If a sequence or collection of points is anomalous with respect to the rest of the data, but not the points themselves, it is referred to as a sequential or collective anomaly. Since this thesis deal with anomalies in time series we will refer to this type of anomaly as a sequential anomaly. An example of a sequence anomaly is if a sensor that records process values fails and from that point outputs the same process value, which is illustrated in Figure 2-6(center). Note that these values are not considered anomalous themselves, but their sequence of them is.
- **Contextual Anomalies**: If a point or a sequence of points are considered as an anomaly with respect to its local neighborhood, but not otherwise, it is referred to as a contextual anomaly. For example, suppose a process can target different qualities at different times resulting in changes of the process values for each quality. Let the qualities result in process target values of three different levels 1, 2 and 3. If the process is running a quality at level 2 and there is an instance of a process value close to those of level 3 this is a contextual anomaly, however, globally a process value close to level 3 is not anomalous when running the quality of that level. This is illustrated in Figure 2-6(right).

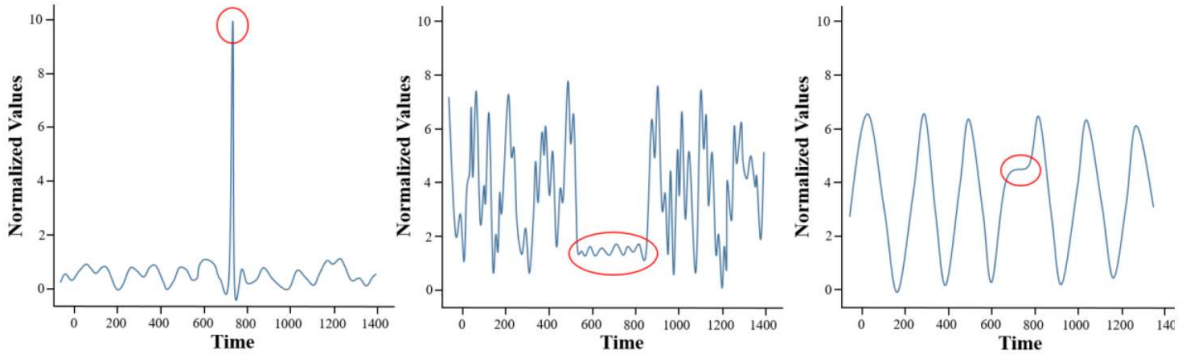


Fig. 2-6 Point anomaly (left), sequential anomaly (center) and contextual anomaly (right)

2.2.2 Data Labels

The labels associated with a data instance denote if that instance is normal or anomalous1. It should be noted that obtaining labeled data that is accurate and representative of all types of behaviors, is often prohibitively expensive. Labeling is often done manually by a human expert and hence requires substantial effort to obtain the labeled training data set. Typically, getting a labeled set of anomalous data instances which cover all possible types of anomalous behavior is more difficult than getting labels for normal behavior. Moreover, anomalous behavior is often dynamic in nature, e.g., new types of anomalies might arise, for which there is no labeled training data. In certain cases, such as air traffic safety, anomalous instances would translate to catastrophic events, and hence will be very rare.

2.2.3 Types of Anomaly Detection

In terms of anomaly detection, there are generally three different mechanisms: supervised approach, semi-supervised approach, and unsupervised approach [2, 7]:

1. Supervised anomaly detection: refers to setups where fully labeled training and validation datasets are available. In this case, the anomaly detection problem degenerates into a classification problem, where the labels of the two categories are often highly imbalanced. However, acquiring thoroughly labeled datasets for anomaly detection is unrealistic due to the following two reasons: 1) data collection and labeling process is time-consuming and labor-

intensive in real-world practice; 2) there will always be potentially unseen novelties in testing or new data.

In other words, since anomalous patterns are in theory countless, it would never be possible to collect all potential features and characteristics of the anomaly class during training.

2. Semi-supervised anomaly detection: approaches that assume the training dataset comprises merely normal (non-anomalous) data [38, 47]. Since only one class is present in training data, the semi-supervised approach in anomaly detection is also known as one-class classification. The basic idea of semi-supervised approaches is to train and validate a model that only fits on normal patterns, and the anomalous data would have a far larger model loss compared to non-anomalous data. Therefore, this loss could be treated as an anomalous score during testing or implementation. Nevertheless, building such a one-class training dataset would still require great efforts and investments [48], which makes such semi-supervised models non-generic.

3. Unsupervised anomaly detection: setups where no assumption of data labels is required [12, 31, 36], which is the most realistic and generic approach in practice. Under some circumstances, semi-supervised anomaly detection could be adapted and transformed to unsupervised anomaly detection by tolerating the minority anomalous samples in training data.

Considering the scope and purpose of this research, this thesis only investigates and studies unsupervised anomaly detection methods. We also include some semi-supervised anomaly detection algorithms that could be transformed into unsupervised anomaly detection models.

An important implicit assumption of unsupervised anomaly detection is:

Assumption 1. Normal records happen far more frequently than anomalous records. Consequently, unsupervised models would only learn normal patterns during training and thus be capable of spotting anomalies during testing.

Apparently, Assumption 1 would not stand when either 1) normal data does not dominantly outnumber anomalies in training data or 2) the unsupervised model does tolerate the rare anomalous pattern and therefore fails to detect similar anomalies in testing.

To mitigate this weak assumption, unsupervised anomaly detection designs are broadly and commonly built upon Encoder-Decoder architecture, where an encoder first generates a latent representation of the input sequence such as sentences, time series, and videos, then a decoder reconstructs another sequence of variables from the encoded data. Since the Encoder-Decoder structure is often applied in sequence modeling and processing, this architecture is also frequently referred to as Sequence to Sequence (seq2seq) models [49], which originates in the natural language processing (NLP) field, and is now widely used in machine translation [50], video captioning, and time series forecasting. Within the scope of this thesis, Encoder-Decoder models and seq2seq models represent the same architectures, where the model outputs are reconstructions of the inputs. In general, seq2seq designs are applied in anomaly detection under the following assumption [39, 47]:

Assumption 2. Only normal patterns could be effectively reconstructed through the model.

Based on Assumption 2, an anomaly scoring mechanism based on intrinsic data patterns could be adopted to distinguish between normal and anomalous data points [7]. Some typical evaluation metrics include loss distances and data densities. For example, under Assumption 2, an Encoder-Decoder model would fit on the data points in Figure 2-5 such that only blue points could be effectively reconstructed. After parameter tuning, the model would be tuned into an estimator of the light blue curve. During testing, we could adopt the l_2 norm with respect to the light blue curve as the anomaly detection metric, and all the points whose l_2 norms are larger than a certain threshold would be classified as anomalous.

2.2.4 Output of Anomaly Detection

An important aspect for any anomaly detection technique is the manner in which the anomalies are reported. Typically, the outputs produced by anomaly detection techniques are one of the following two types:

Scores

Scoring techniques assign an anomaly score to each instance in the test data depending on the degree to which that instance is considered an anomaly. Thus, the output of such techniques is a ranked list of anomalies. An analyst may choose to either analyze top few anomalies or use a cut-off threshold to select the anomalies.

Labels

Techniques in this category assign a label (normal or anomalous) to each test instance. Several techniques, internally, calculate a score for each test instance and use either a threshold or a statistical test to assign a label.

Scoring based anomaly detection techniques allow the analyst to use a domain specific threshold to select the most relevant anomalies. Techniques that provide binary labels to the test instances do not directly allow the analysts to make such a choice, though this can be controlled indirectly through parameter choices within each technique.

Taxonomy

The taxonomy consists of three classes of anomaly detection methods for multivariate time series. These are: conventional approaches, machine learning-based and DNN-based methods.

Conventional approaches, which are also referred to statistical methods by some authors [51], rely on the assumption that a stochastic model generates the observed data and their aim is to estimate a model's parameters from the data and then use the model for prediction [52]. It is often the case that the model hypothesis is considered linear.

The boundary between conventional and machine learning-based approaches is not fully clear. **Machine learning-based models** produce predictions about the results of complex mechanisms by mining databases of inputs for a given problem, without necessarily having an explicit assumption about a model's hypothesis. In this setup, a method aims to learn a function that operates input data to predict output responses [52].

Finally, **DNN-based methods** are a subclass of non-linear machine learning based methods that use neural networks with multiple layers [28].

2.3 Artificial Neural Networks

The field of Artificial Neural Network (ANN) has its origin in neurobiology. The human brain consists of a complex network of approximately 100 billion nerve cells, or neurons, being connected by synapses. In this biological scenario, neurons communicate over the synapses with electrical impulses and a single neuron typically receives many thousands of signals from other neurons. The voltage of an impulse depends on the strength of the actual synapse connection. The total strength of all signals to a neuron can be regarded as the sum of all impulses and each neuron has a threshold mechanism, where signals exceeding it will result in the neuron generating its own voltage impulse[53].

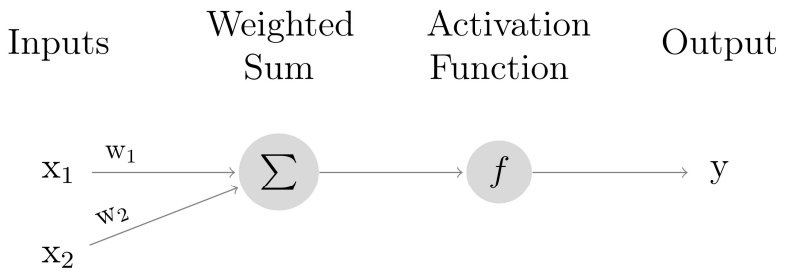


Fig. 2-7 A basic artificial neuron, showing inputs x1 and x2, each paired with respective weight. The activation function node processes a linear combination of x and w, outputting a value based on the function f.

The equivalent functions of the artificial neuron work very similar to the biological and the same glossary is often used in both cases. The principles of an artificial neuron are shown in figure 2-7. The artificial network consists of nodes being interconnected by edges and the strength of the biological synapses is modeled by the edges having a multiplicative weight

factor. The neuron calculates a weighted sum based on all its inputs, resulting in a value that is used in the activation function. The activation function, sometimes also called transfer function, acts as a threshold and there are many different types of functions depending on the desired outcome.

Each hidden unit h calculates a weighted sum a_h of its n inputs and each respective weight w_{ij} . The activation function is then applied to a_h, to calculate the actual output from each unit:

$$a_h = \sum_{i=1}^{n} w_{ih} x_i \tag{2-4}$$

Two of the most common activation functions used in ANNs are the hyperbolic tangent function,

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2-5}$$

limiting all values to [-1, 1], and the logistic sigmoid function $\sigma(x)$ with a range of [0, 1][54].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2-6}$$

A third activation that has become popular in the last few years is the Rectified Linear Unit (ReLU) function,

$$f(x) = \max(0, x) \tag{2-7}$$

ReLU works simply by being a thresholded zero and has been found to accelerate convergence when training ANNs. [55]

In the last layer of the neural network, the output nodes calculate the resulting value of the whole network in the same way as the nodes earlier. However, it is not necessary for the

output nodes to use the same activation function and that choice depends on the task being solved. In the case of a multiclass classification with K classes, a common approach is to apply the softmax function, seen in equation 2-8. The function ensures that the sum of all the outputs is one. [56]

$$f(\mathrm{z})_j = \frac{e^{z_j}}{\sum_{n=1}^{K} e^{z_n}} \quad \text{for} \quad j = 1, \ldots, K \tag{2-8}$$

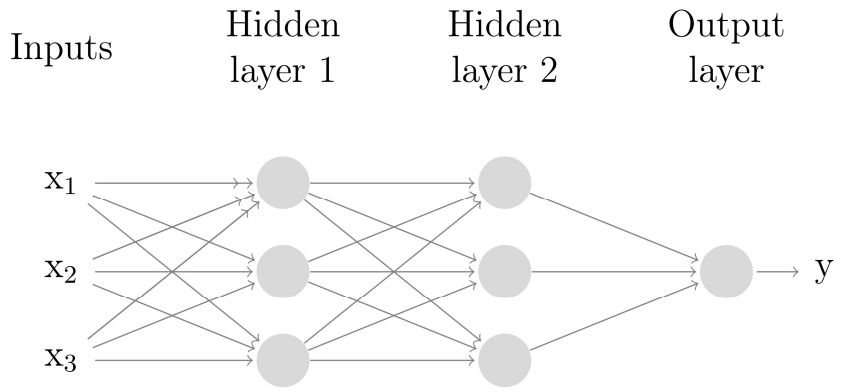


Fig. 2-8 An Artificial Neural Network of feedforward type consisting of an input layer, two hidden layers and a single node as output layer.

Multiple artificial neurons create a Neural Network (NN) and the nodes are commonly structured in layers, as can be seen in figure 2-8. The layers in the network are arranged with input and output layers, with a number of hidden layers in-between. The structure of the connections between layers depends on the type of network; one significant difference is whether connections are forming cycles or not. Feedforward Neural Network (FNN) is an acyclic network and the most widely used type is the Multilayer Perceptron (MLP), which is the type shown in figure 2-8. ANNs consisting of cycles are called recurrent, or feedback, neural networks, and are discussed further in section 2.3.3.

2.3.1 Training an Artificial Neural Network

Training an ANN is performed by exposing the network to typical data and adjusting the weights, such that the correct output can be reproduced given a specific input. The most commonly used procedure is performed in two steps, containing a forward pass and a backward pass. The forward pass consists of processing the input data, as seen in figure 2-7, in each neuron of each layer in the network.

The goal of training the network is to minimize the error between the calculated output $\hat{Y}$ and the target output Y. A commonly used error function is Mean Squared Error (MSE),

$$MSE(Y,\hat{Y}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 \tag{2-9}$$

which is used in cases where the outputs are numerical values. There are cases where the predictions from the model are distributions instead of numerical values, which is the case of the softmax function in equation 2-8. In this case, Categorical Cross Entropy (CCE) is a frequently used error function. CCE is an error function between two distributions Y and $\hat{Y}$, where Y is the true case and $\hat{Y}$ is an approximation of Y. Each distribution consists of a number of probability values, where 0 represents definitely false and 1 definitely true. This type of metric punishes heavily a wrong prediction having a high probability. Categorical Cross Entropy is defined as CCE(Y, $\hat{Y}$) and each distribution p and q have N number of classes.

$$CCE(Y,\hat{Y}) = -\sum_{i=1}^{N} Y_i \cdot \log(\hat{Y}_i) \tag{2-10}$$

While training aims to minimize the error, the risk of only following the least amount of error introduces the risk of overfitting. A model that overfits its training data will have weights so targeted at the specific data of the training, which results in a bad ability to generalize when introduced with slightly different data. To counteract this, a concept called regularization was introduced as a complement to the error function. The regularization term discourages the network to model the data perfectly using too many parameters, by penalizing

the loss with an additional term. The combination of an error function and regularization is often called the objective function, which is the term used in this thesis [57].

Several optimization techniques minimize the objective function; one of the most fundamental is gradient descent. The idea of gradient descent is to use the derivative of the objective functions relative to the weights of the network and adjust the weight with a fixed step size in the negative direction [54].

Backpropagation is a method of calculating the gradient and it is basically just a repeated application of the chain rule, as seen in equation 2-11, working backward from the output through the hidden layers. The notations in the equations are as follows, O is objective function, a is calculated output (as seen in equation 2-4), y is expected output.

$$\frac{\delta O}{\delta a} = \frac{\delta O}{\delta y}\frac{\delta y}{\delta a} \tag{2-11}$$

As seen in equation 2-12, the calculation of the derivatives relative to the weights w_{ij}, which is used in gradient descent. [54]

$$\frac{\delta O}{\delta w_{ij}} = \frac{\delta O}{\delta a_j}\frac{\delta a_j}{\delta w_{ij}} \tag{2-12}$$

2.3.2 Convolutional Neural Network

CNNs are a kind of feedforward neural networks for processing data that has a grid-like topology. For instance, image data are basically multiple channels of 2D pixels, and can be evaluated as 2D grid of pixels. Compared to other neural networks, CNNs adopt convolution operation instead of general matrix multiplication in at least one of their layers.

The convolution between two continuous functions $f(t)$ and $g(t)$ is defined as:

$$(f*g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)\mathrm{g(t}-\tau)\mathrm{d}\tau, \tag{2-13}$$

while the convolution between two discrete functions $f[n]$ and $g[n]$ is:

$$(f*g)[n] \triangleq \sum_{m=-\infty}^{\infty} f[m]\mathrm{g}[n-m]. \tag{2-14}$$

In the case of CNNs, f corresponds to the input, while g is referred to as kernel or filter. The output of the operation is called a feature map. Convolutions are used in CNNs to extract local information and features from data. In particular, each kernel is applied to the entire input, allowing it to look for similar patterns or features regardless of locations or translations. In other words, the kernels of CNNs are location invariant.

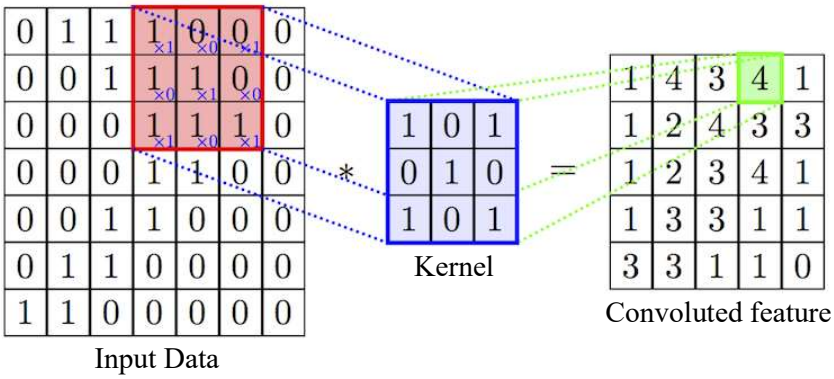


Figure 2-9 Convolution operation in CNNs.

Figure 2-9 shows an example of a convolution operation used in CNNs. The input is a 3 × 3 2D matrix, while the kernel is a 2 × 2 2D matrix. The output of the highlighted sub-matrix is derived as a demonstration, where each individual element within the region is involved in the calculation. The kernel processes all such sub-matrices before outputting the feature map.

Compared to MLP, the weights of parameters in CNNs are shared, which relieves the computation burden caused by potentially large numbers of hidden layers. Moreover, CNNs do not require the input to be flattened into 1D vectors, thus preserving the spatial or high-dimensional information that is otherwise discarded by MLP.

2.3.3 Recurrent Neural Network

As discussed in section 2.3, connections between neurons are never allowed to form cycles in regular feedforward neural networks. This limits the network's ability to make assumptions of relations between data samples because the state of the network is lost after

each sample has been processed. These networks are therefore not as suitable for processing tasks with data sequences related in time or space, such as words in sentences and time series. [58] Taking the example of wanting to predict the next word in a sentence of written text, it is advantageous for the network to consider words that are much earlier in the sentence for a more accurate prediction.

Recurrent Neural Networks (RNN) were introduced with the task of being able to pass along the current state for future sequence steps to use. This may seem like a minor extension of the functionality of the MLP, but the implications are extensive. While basic neural networks are limited to only mapping an input to a corresponding output given a set of weights, the RNN can model whole sequences of dependent items in regard to both input and output. This is turn means that an RNN, theoretically, can model the entire history of previous inputs and outputs [54]. Comparing this to the fixed context window that a regular neural network handles, the strength of the RNN starts to show.

An RNN works particularly well with modeling any type of sequential data, and it is commonly used in word prediction and machine translation applications. Another big use case is in image and video processing, since even inherently non-sequential data, such as a single image, can be represented as a sequence using transformations. [58]

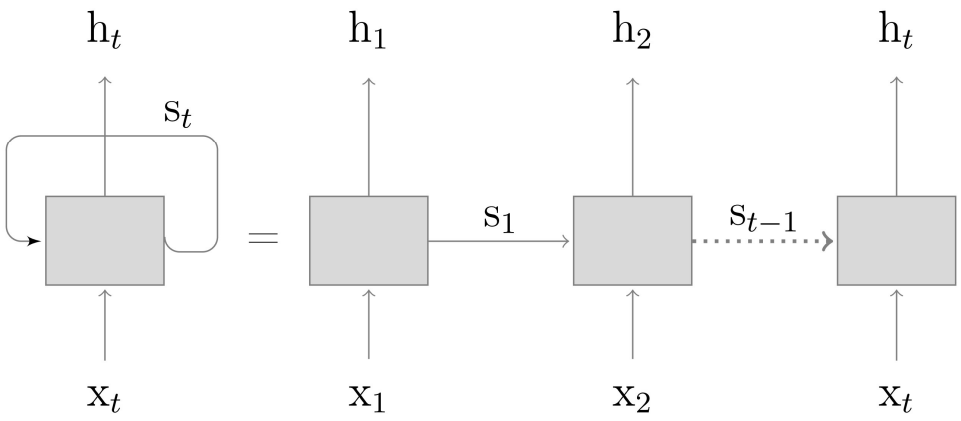


Figure 2-10 Recurrent Neural Network

The recurrent part of the RNN comes from the network performing the same operation for every element of a sequence, having the output from one element as extra input to the next. As can be seen in figure 2-10, a way of visualizing this is to unroll the loop and more clearly show that the network processes the input of each step in the sequence. A sequence containing five words would in this way be shown as a 5-layer network, one layer for each word.

Inspecting a single layer unit, as shown in figure 2-11, shows a single activation function combining the current input and the output from the previous sequence step. The same activation functions can be used in the RNN as in MLP, and an often used function is the hyperbolic tangent function (tanh), which is shown in equation 2-5.

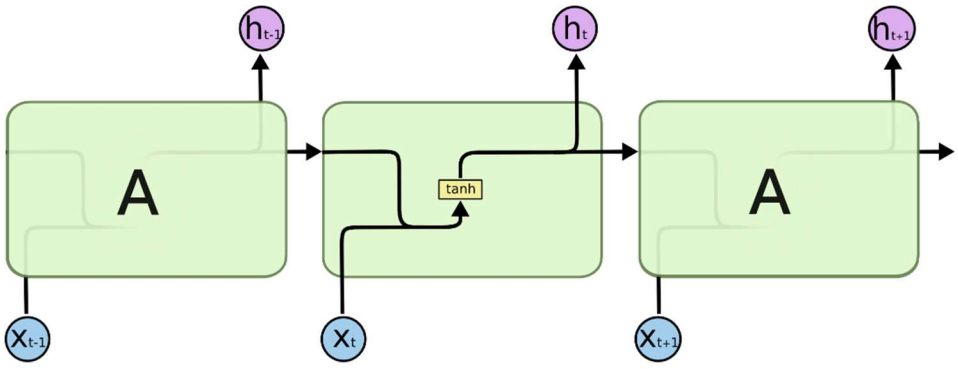


Figure 2-11 In the unrolled visualization of an RNN, each layer unit has an activation function, in this case the hyperbolic tangent function (tanh) [59].

By using the abstraction of unfolding the RNN, it makes it clearer that the same backpropagation procedure, as described in section 3.2, can be used to propagate back across several steps. This algorithm is called Backpropagation Through Time (BPTT) and is essentially the same as regular backpropagation, with the important distinction that the gradients are summed at each step t of the sequence, see equation 2-15. This is relevant in the case of an RNN, since the network passes along parameters across sequence steps, in contrast to a regular ANN. [60]

$$\frac{\delta O}{\delta w} = \sum_t \frac{\delta O_t}{\delta w} \tag{2-15}$$

A negative aspect of the RNN is that, while it can model the dependencies between items in a sequence, it suffers from the difficulty of learning long-range dependencies. This imposes a problem for example when modeling language, since the meaning of a sentence often relates to words that are not close. For example, in the sentence “The man who wore a wig on his head went inside”, the meaning is about the man going inside, not the wig. [61] The underlying problem is called vanishing gradient and relates to the workings of backpropagation, explained in equations 2-12 and 2-15. Due to the way the propagation is a multiplicative operation with the gradients, the contribution of input at time t will be multiplied with an increasingly smaller factor. This results in the gradient shrinking exponentially fast. The problem can also be the opposite, depending on the activation functions, with an exploding gradient, with a gradient so much larger in the earlier layers that others have no effect at all. It is worth noting that neither of these problems is exclusive to the RNN, but they are more apparent compared to a regular FNN due to the design of an RNN being as deep as the sequence length [58, 61].

2.3.4 Long Short-Term Memory

Long Short-Term Memory networks (LSTM) were introduced by Hochreither and Schmidhuber [62] in 1997 as a special type of RNN, aiming to solve the vanishing gradient problem. Having been specifically designed to handle long-term dependencies, LSTMs quickly became popular as an alternative to the RNN in applications such as natural language processing. [59]

As seen in figure 2-12, the structure of the LSTM can be visualized in an unfolded manner similarly to the RNN in figure 2-11. Where the RNN functions with a single layer, the LSTM has four co-operating layers. The main addition introduced in the LSTM was the cell state C, which acts as a memory channel. This means that, instead of a single output, two

outputs, o_t and h_t, are calculated per step. These are affected by the four layers in different ways, as described below [59].

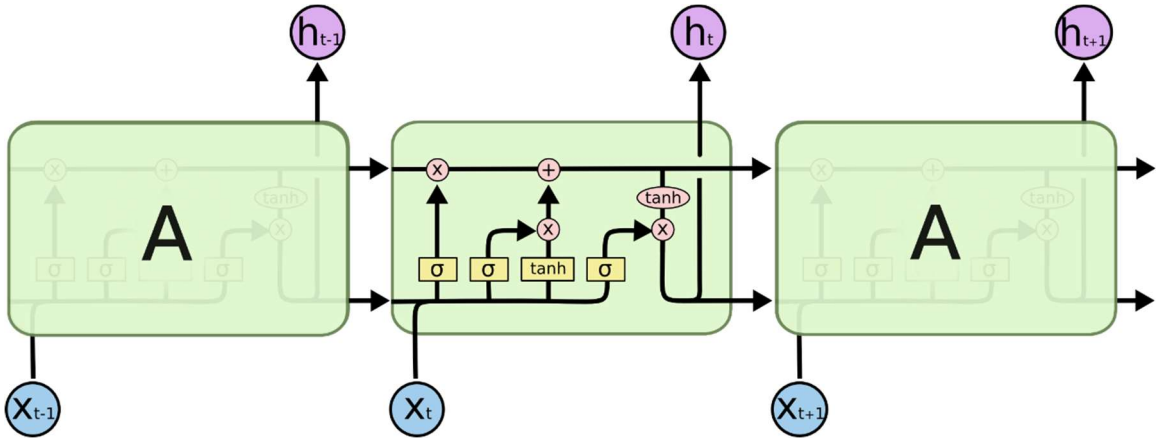


Figure 2-12 Structure of an LSTM memory [59]

The first sigmoid (σ) layer acts as a "forget gate", taking both the previous output h_{t-1} and current input x_t into consideration when deciding how much of the cell state C_{t-1} should be remembered. The result is f_t, as shown in equation 2-16, where $f_t = 1$ keeps C_{t-1} as it is and $f_t = 0$ completely disregards it.

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t + b_f) \tag{2-16}$$

The next two layers act together deciding the information that will be added to the cell state. The first part is a sigmoid layer, which calculates a vector using equation 2-17 deciding how much of each state value that should be updated. The second layer is a tanh layer, which bears a resemblance to the single layer of the RNN as seen in figure 2-11. As shown in equation 2-18, this layer calculates values that potentially could be important to store in the cell state.

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t + b_i) \tag{2-17}$$

$$\tilde{C}_t = \tanh(W_{hC} h_{t-1} + W_{xC} x_t + b_C) \tag{2-18}$$

The actual update of the cell state is performed with an addition operation between the candidate values calculated in $i_t * \tilde{C}_t$ and the current cell state, as shown in equation 2-19. Performing this as an additional means that new information can unobstructedly be added to the cell state.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2-19}$$

The final part of the LSTM block calculates the output of this step t. As shown in equations 2-20 and 2-21, this is performed in two steps. A sigmoid layer is yet again used as a masking vector o_t, using information in the input to decide what parts of the cell state that is going to be outputted. The cell state is used together with a basic activation function tanh and then combined with o_t, resulting in the output h_t consisting only of the parts that are calculated to be significant.

$$o_t = \sigma(W_{ho} h_{t-1} + W_{xo} x_t + b_o) \tag{2-20}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{2-21}$$

2.4 Autoencoders

An auto-encoder [28, 32] is a type of neural network used in unsupervised learning in which the network is composed of an encoder and a decoder sub-models. The encoder forces a compressed representation of the input in smaller dimensions and the decoder attempts to recreate the input from the compressed version provided by the encoder.

Auto-encoders are applied to many problems, from facial recognition, feature detection, and data denoising. They represent data within multiple hidden layers by reconstructing the input data, effectively learning an identity function. When trained solely on normal data instances, they fail to reconstruct the anomalous data samples producing a large reconstruction error. These points associated with a high residual error are considered anomalies.

The choice of autoencoder architecture depends on the nature of data, convolutional networks are preferred for image datasets while Long short-term memory (LSTM) based models are able to capture the time dependency in sequential data.

The deep of an autoencoder depends on the dimension of the input data. The more dimensions, the more layers are needed to extract all the relevant information during training.

The type of learning is unsupervised because the model does not require any information about the labels, making it very popular and widely used in literature.

The encoder takes the input data $x \in \mathbb{R}^{d_x}$ to a latent space(code) $z \in \mathbb{R}^{d_z}$.

$$z = \sigma_1(\mathrm{W}x + \mathrm{b}_1) \tag{2-22}$$

where z is latent space or code, σ_1 is an activation function, $\mathrm{W} \in \mathbb{R}^{d_x \times d_z}$ is a weight matrix and $\mathrm{b}_1 \in \mathbb{R}^{d_z}$ is a bias vector of an encoder.

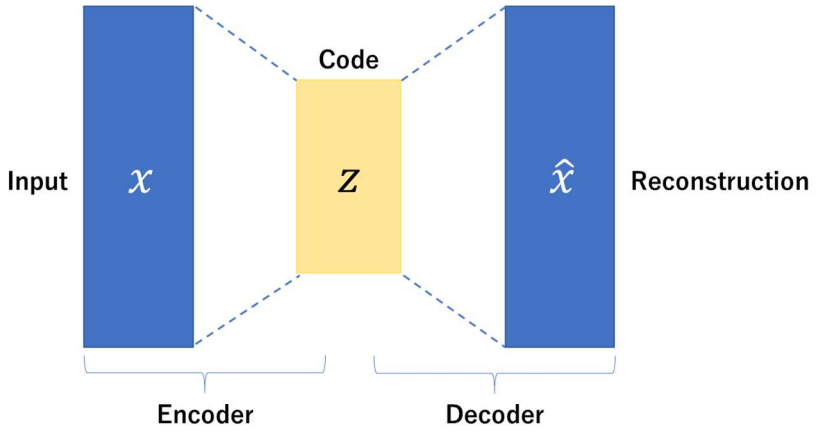


Fig. 2-13 The architecture of an autoencoder

Autoencoders consist of an encoder and a decoder. The encoder takes the input data $x \in \mathbb{R}^{d_x}$ to a latent space(code) $z \in \mathbb{R}^{d_z}$.

$$z = \sigma_1(\mathrm{W}x + \mathrm{b}_1) \tag{2-23}$$

where z is latent space or code, σ_1 is an activation function, $\mathrm{W} \in \mathbb{R}^{d_x \times d_z}$ is a weight matrix and $\mathrm{b}_1 \in \mathbb{R}^{d_z}$ is a bias vector of an encoder.

The decoder maps z to the reconstruction $\hat{x}$ of the same dimension as x.

$$\hat{x} = \sigma_2(\mathrm{W}'\mathrm{z} + \mathrm{b}_2) \tag{2-24}$$

where $\hat{x}$ is the reconstruction or output of the autoencoder, σ_2 is an activation function of the decoder, $\mathrm{W}' \in \mathbb{R}^{d_x \times d_z}$ is a weight matrix and $\mathrm{b}_2 \in \mathbb{R}^{d_x}$ is a bias vector of a decoder.

The autoencoders are trained to minimize a reconstruction loss function $L(x, \hat{x})$, which measures how well the decoder performs and how the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.

The Mean Squared Error (MSE) is the most common loss function.

$$L(x, \hat{x}) = \left\| x - \hat{x} \right\|^2 \tag{2-25}$$

$$L(x, \hat{x}) = \left\| x - (\sigma_2(\mathrm{W}'(\sigma_1(\mathrm{W}x + \mathrm{b}_1)) + \mathrm{b}_2)) \right\|^2 \tag{2-26}$$

Autoencoders can be under complete which is one of the simplest types of autoencoders, i.e., under complete autoencoders latent code z has a lower dimensionality than the input space x, which is forced to learn a compressed representation of the data. An autoencoder can be used for dimensionality reduction tasks in this framework. If the autoencoder has just one hidden layer and if the functions are linear and the loss is the mean squared error, an autoencoder is provably equivalent to Principal Component Analysis (PCA), while the weights to the K hidden units will span the same subspace as the first K principal components of the data. Furthermore, if the activation functions are non-linear, autoencoders can find non-linear representations of the data and, therefore, they are a powerful generalization of PCA that has experimentally demonstrated impressive results in the past.

2.5 Variational Autoencoders

Autoencoders were traditionally mainly used for dimensionality reduction and representation learning. More recently, theoretical connections to latent variable models have resulted in the variational autoencoder (VAE) [63]. VAE is a generative model with a probabilistic background, which can exploit its ability to model very complex distributions in a latent space in order to detect anomalies.

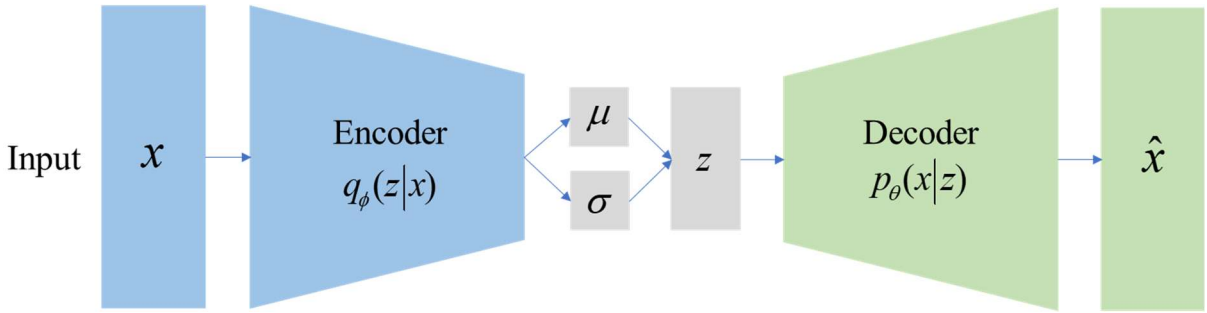


Fig. 2-14 The architecture of a Variational autoencoder

Probabilistic latent variable models

Probabilistic Latent Variable Models (LVM) constitute a broad class of explicit generative models, they represent a common approach to the unsupervised representation learning problem. LVMs utilize auxiliary variables to express complex distributions that seize more realistically natural aspects of the Universe. The observed variable x is supposed to be generated by a stochastic process based on an unobserved continuous variable z: firstly, the hidden z is generated from a prior distribution $p^*(z)$, then x is generated from a conditional distribution $p^*(x|z)$. The unobserved variable z can be interpreted as a latent representation. Generally, the goal is twofold: to model the joint distribution $p_\theta(x,z)$ and to infer the $p_\theta(z|x)$ distribution for representation learning by employing a LVM, which can be thought of as a simple directed graphical model [64], as illustrated on Figure 2-15. The LVMs hold the

potential to automatically discover the underlying generative process and yield interpretable latent representations that reflect the true generative factors of a particular phenomenon.

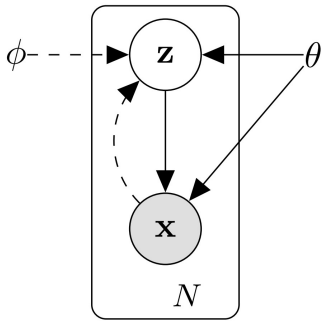


Fig. 2-15 The VAE as a graphical model

The main difference between other explicit density models and the probabilistic latent variable models lies in the posterior distribution over the latent variables $p_\theta(z|x)$, derived from Bayes' theorem. This distribution expectedly lies on a low dimensional manifold that can provide insights into the internal representation of the data [65]. The further motivation behind introducing a hidden variable is that the joint distribution can be defined as a product of simpler distributions using the law of total probability: $p_\theta(x,z) = p_\theta(x|z)p_\theta(z)$. The prior distribution over latent variable z is usually predefined, significantly simplifying the computations required for likelihood estimation.

Maximum Likelihood - Learning from observed data

One convenient procedure to train a generative model on a given dataset $X = \{x^{(i)}\}_{i=1}^{N}$ and find a suitable parametrization of the model distribution $p_\theta(x)$ is to use Maximum Likelihood Estimation (MLE). MLE determines an optimal θ^* parameter. If it exists under which the likelihood of each datapoint from X is as high as possible. This also means placing more probability mass around the regions of the input space containing more samples from X. For computational simplicity and numerical stability, maximizing the logarithm of likelihood function $\ell(\theta; X) = \log p_\theta(\mathrm{X})$ is more favorable than the likelihood $p_\theta(X)$, since likelihoods,

being products of the probabilities of many data points, tend to be very small. Consequently, the generative model parameters are sought as $\theta^* \leftarrow \underset{\theta}{\operatorname{argmax}} \log p_\theta(X)$.

Since it is assumed that the training dataset consists of independent, identically distributed observations, maximizing the log-likelihood of the data is equivalent to maximizing the log-likelihood of each individual data point separately:

$$\log p_\theta(\mathrm{X}) \;=\; \log[p_\theta(x^{(1)}) \ldots p_\theta(x^{(N)})] \;=\; \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$

MLE of θ requires the maximization of the sum, or equivalently the average of log-likelihoods assigned to the training data points, which gives an estimation of $\mathbb{E}_{x \sim p^*(x)} \log p_\theta(x)$. For the efficient optimization of this MLE objective, the Stochastic Gradient Descent (SGD) is used in the field of deep learning, where large datasets are processed [66]. Stochastic Gradient Descent randomly draws mini-batches from X and estimates the gradients of the objective with respect to θ using the data points of a single mini-batch M: $\frac{1}{|X|} \nabla_\theta \log p_\theta(X) \approx \frac{1}{|M|} \sum_{x^{(i)} \in M} \nabla_\theta \log p_\theta(x^{(i)})$. The gradient estimation is used then to iteratively perform gradient descent to reach a local minimum of the negative MLE objective function.

Intractable distributions

In the interest of maximizing the log-likelihood of the training data, it is enough to be able to calculate the gradient of the log-likelihood of a single observation. However, in the case of LVMs, when a continuous latent variable z is brought in, the marginal likelihood of observation becomes intractable since it would require marginalization over the continuous variable z: $\log p_\theta(x^{(i)}) \;=\; \log \int p_\theta(x^{(i)} | z)\, p_\theta(z)\, dz$.

Taking the gradient of $\log p_\theta(x^{(i)})$ is typically infeasible as it depends upon the evaluation of the integral, and the analytic solution is not directly available in the case of raw datasets, nor an efficient estimation in general case.

Thus, the MLE objective cannot be optimized directly. Although one can observe that the integral does not necessarily have to be calculated over all the values of z: it would be enough to evaluate it over z values to which the training data point is mapped with high probability, in other words where $p_\theta(z|x^{(i)})$ is significantly greater than 0. Unfortunately, the $p_\theta(z|x^{(i)})$ likelihood is also intractable considering that $p_\theta(z|x^{(i)}) = \frac{p_\theta(x^{(i)},z)}{p_\theta(x^{(i)})}$, and the denominator is intractable.

Evidence Lower Bound

A subtle idea to circumvent the intractability of the $p_\theta(z|x^{(i)})$ distribution lies within variational Bayesian inference [64]. The key idea taken from the variational Bayesian approach is to approximate the true posterior distribution $p_\theta(z|x^{(i)})$ with a variational distribution $q_\phi(z|x^{(i)})$ defined by an inference model $q_\phi(z|x^{(i)})$ that can be an arbitrary function parametrized with ϕ variational parameter shared across each data point. Amortizing the variational parameters over the entire input allows us to scale this approach even to large datasets, though this will result in a larger gap between the true and modeled log-likelihood [67]. Finding a suitable variational parameter ϕ results in an additional optimization problem that requires the minimization of the distance between the true and approximate posterior distribution. The Kullback-Leibler divergence (KL-divergence) serves as one means to quantify how close two distributions are.

Two noteworthy properties of KL-divergence are its' non-negativity and asymmetricity. The below used form known as reverse KL-divergence is chosen in variational inference not only for the reason that it results in computational simplifications but also indicates preferring an approximation where regions of high $q_\phi(z|x^{(i)})$ are accurate, rather than regions of high $p_\theta(z|x^{(i)})$. This is desire able when drawing samples from $q_\phi(z|x^{(i)})$; however, the resulting approximation usually underestimates the support of the true posterior.

The KL-divergence between the true posterior $p_\theta(z|x^{(i)})$ and the approximate posterior $q_\phi(z|x^{(i)})$ can be written as:

$$D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z|x^{(i)})) = \int_{-\infty}^{\infty} q_\phi(z|x^{(i)}) \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})} dz =$$

$$= \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log q_\phi(z|x^{(i)}) - \log p_\theta(z|x^{(i)})] \quad (2\text{-}27)$$

Rewriting Equation 2-27 using Bayes' theorem reveals that the log-likelihood of a single datapoint can be related to the KL-divergence between true and approximate posterior distributions:

$$D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z|x^{(i)})) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log q_\phi(z|x^{(i)}) - \log \frac{p_\theta(x^{(i)}, z)}{p_\theta(x^{(i)})}\right] =$$

$$= \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log q_\phi(z|x^{(i)}) - \log p_\theta(x^{(i)}, z)] + \log p_\theta(x^{(i)}) \quad (2\text{-}28)$$

Rearranging Equation 2-28 yields the following:

$$\log p_\theta(x^{(i)}) - D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z|x^{(i)})) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\left[\frac{\log p_\theta(x^{(i)}|z) p_\theta(z)}{\log q_\phi(z|x^{(i)})}\right] \quad (2\text{-}29)$$

In Equation 2-29 the KL-divergence term on the left-hand side is still intractable due to the fact that $p_\theta(z|x^{(i)})$ cannot be evaluated . Luckily the non-negativity property of the KL-divergence can be exploited here to discard that term and, in such a way get the Evidence Lower Bound (ELBO):

$$\log p_\theta(x^{(i)}) \geq \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)})] = \mathcal{L}(\phi, \theta; x^{(i)}) \quad (2\text{-}30)$$

It is worthy of note that the gap between the marginal log-likelihood of a datapoint and ELBO is exactly the KL-divergence between the true posterior and the approximation of it, as a consequence, the lower bound becomes tighter as the approximation is improved.

The ELBO can be written in other insightful forms too, as formulated by [Hoffman and Johnson, 2016], which contribute to a better understanding and interpretability. One such version is obtained by reformulating expectation terms as a KL-divergence:

$$\mathcal{L}(\phi, \theta; x^{(i)}) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log(p_\theta(x^{(i)}|z) + \log p_\theta(\mathrm{z}) - \log q_\phi(z|x^{(i)})] =$$

$$= \mathbb{E}_{z\sim q_\phi(z|x^{(i)})} \log(p_\theta(x^{(i)}|z) + \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log p_\theta(\mathrm{z}) - \log q_\phi(z|x^{(i)})] =$$

$$= \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log(p_\theta(x^{(i)}|z)] - D_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z)) \qquad (2\text{-}31)$$

This variant of ELBO clarifies the connection of VAE with traditional autoencoders and helps to simplify the computation needed for training, as further discussed later.

Henceforward the optimization of MLE objective can be replaced with the maximization of the sum of individual-datapoint ELBOs with regard to both θ and ϕ. This objective simultaneously maximizes the marginal likelihood and minimizes the KL-divergence of approximate and true posterior, therefore improving on both the generative and the inference model. The maximization of this objective still depends upon the generally intractable ELBO and its gradients taken with regard to both parameters, although easily calculated unbiased estimators of them exist.

The estimation of gradients with regard to the generative model parameters can be obtained effortlessly as Monte Carlo estimate of expectation using a single sample z:

$$\nabla_\theta \mathcal{L}(\phi, \theta; x^{(i)}) = \nabla_\theta \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)})] =$$

$$= \mathbb{E}_{z\sim q_\phi(z|x^{(i)})} \nabla_\theta [\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)})] \approx \nabla_\theta \log p_\theta(x^{(i)}, z) \qquad (2\text{-}32)$$

A similar estimation of gradients with regard to the variational parameters cannot be calculated since the ELBO's expectation is taken with regard to the approximate posterior distribution depending on ϕ, therefore, the operations of taking the gradient and the expectation cannot be interchanged.

The Reparameterization Trick

To rise above the mentioned problem, a change of variables has to be applied, also known as the reparameterization trick. The concept is to express the random variable z as a deterministic, differentiable transformation of another random variable ε, given $x^{(i)}$ and ϕ: $z = g(\varepsilon, \phi, x^{(i)})$, where the distribution of ε is independent from $x^{(i)}$ and ϕ. Under this reparameterization, the expectation with regard to the approximate variational distribution can be replaced with one with regard to $p(\varepsilon)$ independent of variational parameters. The ELBO can be rewritten then as:

$$\mathcal{L}(\phi, \theta; x^{(i)}) = \mathbb{E}_{\varepsilon \sim p(\varepsilon)}[\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)})], \text{ where } z = g(\varepsilon, \phi, x^{(i)})$$

As a result of reparameterization, the randomness in z is externalized and the ELBO can be straightforwardly differentiated with regard to both θ and ϕ. Likewise, in Equation 2-32 a simple Monte Carlo estimator of the gradients of ELBO can be formulated with the use of a single noise sample $\varepsilon \sim p(\varepsilon)$ and $z = g(\varepsilon, \phi, x)$:

$$\nabla_\theta \mathcal{L}(\phi, \theta; x^{(i)}) \approx \nabla_{\theta,\phi}[\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)})]$$

With the gradient estimates, the SDG can be used to maximize the ELBO objective just like in the case of MLE, but the prior over the latent variables have to be defined. The choice of prior and assumptions regarding the posterior distributions further simplifies the optimization of the ELBO objective.

Stochastic optimization of the ELBO

The optimization of the evidence lower bound was performed using a stochastic optimization procedure proposed in the VAE original paper by Kingma and Welling [63]. They called this algorithm AutoEncoding Variational Bayes (AEVB).

Algorithm 1 Auto-Encoding Variational Bayes Algorithm

Input:

X: Dataset

$q_\phi(z|x)$: Inference model

$p_\theta(x, z)$: Generative model

Output:

θ, ϕ: Learned parameters

$(\theta, \phi) \leftarrow$ Initialize parameters

while SGD not converged **do**

$\mathcal{M} \sim X$ (Random minibatch of data)

$\varepsilon \sim p(\varepsilon)$ (Random noise for every data point within minibatch $\mathcal{M}$)

Compute $\mathcal{L}(\theta, \phi, \mathcal{M}, \varepsilon)$ and its gradients $\nabla \mathcal{L}(\theta, \phi, \mathcal{M}, \varepsilon)$

Update θ and ϕ using a SGD optimizer

end

The stochastic optimization procedure in the AEVB algorithm is two-fold since the noise is introduced by the random choice of a mini-batch $\mathcal{M}$ and by the sampling step $\varepsilon \sim p(\varepsilon)$. Given a dataset $X = \left\{ x^{(n)} \right\}_{n=1}^{N}$ composed by N independent and identically distributed examples, the global evidence lower bound objective is the sum of the evidence lower bounds of all individual data points $x^{(n)}$ within X.

Choice of prior and conditional distributions

The prior $p_\theta(z)$ over the latent variables should be a simple distribution from which one can easily sample, therefore it is usually chosen to be a centered isotropic multivariate Gaussian distribution. With this choice, it will be independent of the ℓ parameters: $p_\theta(z) = p(z) = N(z; 0, I)$. The intractable true posterior $p_\theta(z|x^{(i)})$ is supposed to take on an approximate Gaussian form with approximately diagonal covariance. In this case, the approximate posterior $q_\phi(z|x^{(i)})$ can be chosen to be a multivariate Gaussian distribution with a diagonal covariance $N(z; \mu_\phi, \sigma_\phi^2 I)$, with parameters μ_ϕ and σ_ϕ calculated as functions of a

datapoint $x^{(i)}$ depending on the variational parameters ϕ. The reparameterization of z in this case is simply $z = \mu_\phi + \sigma_\phi \cdot \varepsilon$, where $\varepsilon \sim N(0, I)$.

With these choices of distributions, the KL-divergence term of ELBO in Equation 2-31 takes a closed form, and its gradients can be easily calculated, only the gradients of the other term have to be estimated:

$$D_{KL}(N(z; \mu_\phi, \sigma_\phi^2 I) || N(z; 0, I)) = \frac{1}{2} \sum_{j=1}^{J} [1 + \log((\sigma_{\phi j})^2) - (\mu_{\phi j})^2 - (\sigma_{\phi j})^2].$$

In the above equation J notes the dimensionality of the latent space.

The $p_\theta(x|z)$ noise model is usually also fixed to be a multivariate Gaussian distribution $N(z; \mu_\theta, \sigma_\theta^2 I)$ (or Bernoulli in case of binary input data), whose parameters μ_θ and σ_θ are computed as a function of a single $z^{(i)}$ and depend on θ.

With the choice of Gaussian distribution as noise model, usually the σ_θ variance is fixed to be 1, resulting that the approximation of the expectation term of ELBO in Equation 5 can be written as: $\frac{1}{2} \log 2\pi + ||x^{(i)} - \mu_\theta||_2$, and its gradients can be calculated easily.

Bringing in neural networks

Up to this point, it was not necessary to concretize the exact form of $p_\theta(x|z)$ and $q_\phi(z|x)$, the generative and the inference model. It would be desirable to model $p_\theta(x|z)$ with a parameterized distribution flexible enough to capture the true data distribution, and to approximate $p_\theta(z|x)$ with $q_\phi(z|x)$ well enough. Arbitrary differentiable functions could model conditional probability distributions. From the mathematical theory of artificial neural networks, it is known that neural networks with suitable activation functions are universal approximators, they can approximate any continuous function to any desired precision [68]. This offers a good choice for parametrizing the two conditional distributions with neural networks, allowing for probabilistic reasoning about autoencoder-based generative models.

The resulting VAE can be viewed as a traditional autoencoder with an additional specific regularization term. In the VAE setup, the inference model $q_\phi(z|x)$ takes the role of an encoder responsible for stochastically mapping the input data points to latent representations, and similarly, the generative model $p_\theta(x|z)$ acts as a probabilistic decoder.

The encoder and decoder are jointly trained to maximize the log-likelihood of each training data point through the minimization of the VAE objective function:

$$\mathcal{L}(\phi, \theta; X) = \sum_{i=1}^{N} \left(\mathbb{E}_{z \sim q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)}|z)] - D_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z)) \right)$$

In the individual-datapoint ELBO, Equation 2-31, the first term is the distortion term quantifying the reconstruction error of observation, the objective minimized by autoencoders, while the second KL-divergence is the rate term that measures the additional number of extra bits required to encode a sample from the true posterior using a code optimized for encoding samples from the variational approximation of the prior [69]. The KL-divergence term can be interpreted as a regularizer that is minimized when $q_\phi(z|x) = p_\theta(z)$ for all *x*. This perspective has been used to explain the tendency of VAE to discard the majority of dimensions in latent z leading to the posterior collapse problem discussed in the next subsection along with other drawbacks and advantages of this model.

Advantages and Disadvantages of Variational Autoencoders

The main advantages of VAEs rely on algorithms for unsupervised learning. Unsupervised learning is the natural procedure that cognitive mammals, i.e., human beings, use for learning, which makes it an interesting alternative for machine learning and artificial intelligence. This consists of the network discovering the data features on its own and using those features to classify the data later. In this way, there is no need to define an input and output dataset beforehand, like in supervised learning.

It was also mentioned that VAEs have simple structures, which is an advantage compared to Generative Adversarial Networks. In this way, they are easier to train, joint with the fact that VAEs have a clear objective function to optimize (log-likelihood).

Another advantage that variational autoencoders present against GANs is that the quality of their models can be evaluated by means of the log-likelihood (explained in the following sections). At the same time, GANs cannot be compared except by visualizing the samples.

However, VAEs present a drawback in terms of reconstruction since the generated images are blurred when compared to the ones generated by GANS. This blurred is caused by the imperfect reconstruction achieved by variational autoencoders.

2.6 Moving Average

2.6.1 Exponentially Weighted Moving Average

The Exponentially Weighted Moving Average (EWMA) is a quantitative or statistical measure used to model or describe a time series. The EWMA is widely used in finance, the main applications being technical analysis and volatility modeling.

The moving average is designed such that older observations are given lower weights. The weights fall exponentially as the data point gets older; hence the name is exponentially weighted.

The only decision a user of the EWMA must make is the parameter alpha. The parameter decides how important the current observation is in calculating the EWMA. The higher the alpha value, the more closely the EWMA tracks the original time series. The EWMA's simple mathematical formulation is described below:

$$\mathrm{EWMA}_t = \alpha * x_t + (1 - \alpha) * \mathrm{EWMA}_{t-1} \tag{2-33}$$

where EWMA_t is moving average at time t, α is a degree of mixing parameter value between 0 and 1, and x_t is the value of the series in the current period.

This formula states the value of moving average at time t. Here is a parameter that shows the rate at which the older data will come into calculation.

If α=1, that means only the most recent data has been used to measure EWMA. If α is nearing 0, that means more weightage is given to older data, and if α is near 1, that means newer data has been given more weightage.

2.6.2 Linearly Weighted Moving Average

Linearly Weighted Moving Average (LWMA), also referred to as a weighted moving average, LWMA is a simple moving average that places more weight on recent data. The most recent observation has the biggest weight and each one prior to it has a progressively decreasing weight.

$$\mathrm{LWMA}_t = \frac{(x_t * n)+(x_{t\text{-}1} * (n\text{-}1))+\ldots+(x_{t\text{-}n+1} * (1))}{n(n+1)/2} \tag{2-34}$$

where LWMA_t is the value of the current period at time t, n is the number of periods, and x_t is the value of the series in the current period.

2.7 Attention Mechanism

The idea of integrating attention in neural network models is partially inspired by the human attention system that has the ability to select stimuli during the early stages of processing based on elementary stimulus features [70]. An interesting example is the human visual system that can selectively focus its attention on parts of the visual space in order to acquire information when and where it is required and build its own representation of the scene.

Sequence to Sequence models have their weakness in tackling long sequences (e.g., long time series), mainly because the intermediate fixed-length vector representation does not have enough capacity to capture information from the entire input sequence, x. In other words, longer sequences need to be encoded into the same fixed-length vector representation or context vector.

Rooted in the mechanism behind the human attention system, Attention Mechanisms (AM) were proposed to overcome this limitation by allowing the decoder to attend to relevant encoded hidden states selectively.

Several attention models have been proposed in the past few years [71, 72], and, in general, they operate as follows. At each timestep t, during decoding, the attention model computes a context vector obtained by a weighted sum of the encoder's hidden states.

Even though attention was developed mainly in the framework of Natural Language Processing (NLP) tasks involving text data, it can be applied to other problems dealing with other types of data, such as time series and videos. Attention mechanisms have shown an impressive success in a variety of applications such as machine translation [71], image classification [73], speech recognition [74], pose estimation [75], sentence summarization [76] and image captioning [77]. In fact, attention is a natural extension of approaches based on Seq2Seq models for any kind of sequential data.

2.8 Evaluation metrics

There are different methods and metrics that can be of use when evaluating the performance of a classifier. In unsupervised learning, the evaluation is heavily dependent on the problem at hand and is often a complicated task. An anomaly threshold can be set as a decision boundary in anomaly detection so that the algorithm can be evaluated as a regular classifier. The classification performance can then be measured for different threshold values to see various classifier performance attributes as the threshold increases or decreases.

The confusion matrix is commonly used to show how each test value predicts classes compared to their actual classes. It is a table with four different combinations of detected and actual values: True Positive or TP, False Positive or FP, True Negative or TN, and False Negative or FN. In this way, we can assign the anomalies as positive and the normal as negative in the anomaly detection task. The definitions of TP, FP, TN, and FN are described as follows:

True Positives (TP): The cases are abnormal and detected as anomalies.

True Negatives (TN): The cases are normal and detected as normal.

False Positives (FP): The cases are normal but detected as anomalies.

False Negatives (FN): The cases are abnormal but detected as normal.

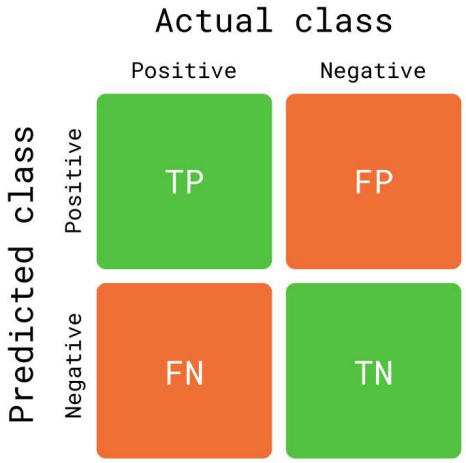


Fig. 2-16 A confusion matrix

Furthermore, to evaluate the anomaly detection model, a few evaluation metrics can help calculate anomaly detection based on the confusion matrix: precision, recall, F1-score, True Positive Rate (TPR), and False Positive Rate (FPR).

Precision is the ratio of correctly detected anomalies to all detected anomalies. Mathematically this is defined by the equation below.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2-35}$$

The recall is the ratio of correctly detected anomalies to all anomalies.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2-36}$$

F1-score combines recall and precision into one performance metric. F1-score is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account. F1-score is very useful, especially in imbalanced data.

$$\text{F1}-\text{Score}= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2-37}$$

TPR is the same as recall.

$$\text{TPR} = \text{Recall} = \text{Sensitivity} = \frac{TP}{TP+FN} \tag{2-38}$$

FPR shows the proportion of wrongly judged anomalies in all normal samples. The formulas of these metrics are shown below.

$$\text{FPR} = \text{1-Specificity} = \frac{FP}{FP+TN} \tag{2-39}$$

A receiver operating characteristic (ROC) curve is a plot that illustrates the performance of a classification model for all threshold settings. FPR represents the x-axis, and TPR represents the y-axis. As the threshold is altered in both directions, the FPR and TPR will range between zero and one. As the ROC curve has been constructed, the area under the ROC curve (AUROC) can be measured. In cases where the real function is unknown, and the integral can not be directly calculated, approximation methods such as the trapezoidal rule can be used. The formal definition of the trapezoidal rule can be seen in Equation 2-37. An AUROC of 0.0 reflects the model predicting the wrong output at all times, an AUROC of 0.5 means the model has not learned anything useful, and an AUROC of 1.0 means the model is optimal at all threshold values.

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2}(x_k - x_{k-1}) \tag{2-40}$$

where: f = approximated function

x = data points

2.9 Deep learning anomaly detection methods (DNN)

DNN-based methods are a sub-category of machine learning-based approaches, which rely on deep neural networks. Given the explosion of DNN-based methods over the last years, they are presented as a separate category.

Auto-Encoder (AE) [32] is an artificial neural network combining an encoder and a decoder. The encoder part takes the input window and maps it into a set of latent variables z, whereas the decoder maps the latent variables z back into the input space as a reconstruction. The difference between the original input vector and the reconstruction is called the reconstruction error. Thus, the training objective aims to minimize this error. Auto-encoder-based anomaly detection uses the reconstruction error as the anomaly score. Time windows with a high score are considered to be anomalies.

Generative Adversarial Networks (GANs) [66] have the ability to know whether an input sample is normal or not. A GAN is an unsupervised artificial neural network based on a two-player minimax adversarial game between two networks, which are trained simultaneously. One network, the generator (G), aims to generate real data, whereas the second one acts as a discriminator (D) trying to discriminate real data from that one generated by G. The training objective of G is to maximize the probability of D making a mistake, whereas the training objective D is to minimize its classification error. Similarly to AE-based, GAN-based anomaly detection uses normal data for training. After training, the discriminator is used as an anomaly detector. If the input data is different from the learned data distribution, the discriminator considers it as coming from the generator and classifies it as fake, i.e., as an anomaly.

The Long Short-Term Memory Variational Auto-Encoders (LSTM-VAE) [38] combines the LSTM [62] which is a recurrent neural network architecture with a variational auto-encoder (VAE) by replacing the feed-forward network in a VAE with a long short-term

memory (LSTM). The LSTM-VAE models the time dependence of time series through LSTM networks. During encoding, the LSTM-VAE projects the input data and its time dependencies into a latent space. It uses the latent space representation to estimate the output distribution during decoding. Finally, the LSTM-VAE detects an anomaly when the log-likelihood of the current data is below a threshold. S. Lin et al. [78] show that the LSTM-VAE is capable of identifying anomalies that span over multiple time scales.

The Deep Autoencoding Gaussian Mixture Model (DAGMM) [31] jointly considers a Deep Auto-encoder and a Gaussian Mixture Model (GMM) to model the density distribution of multidimensional data. The Deep Autoencoder aims to generate a low-dimensional representation and a reconstruction error for each input data time window. This representation is used as input of a Gaussian Mixture Model (GMM). The parameters of the Deep Auto-encoder and the mixture model are optimized simultaneously from end to end, taking advantage of a separate estimation network to facilitate the learning-based of the parameters of the mixture model. The DAGMM then uses the likelihood to observe the input samples as an anomaly score.

The Multivariate Anomaly Detection with Generative Adversarial Networks (MAD-GAN) [37] is based on a Generative adversarial network (GAN) [66] architecture composed of LSTMs. MAD-GAN uses an anomaly score called DR-score to detect anomalies. This score is composed of the discrimination between real data and fake data of the discriminator and the reconstruction error of the generator. Indeed, because of the smooth transitions of the latent space, the generator produces similar samples if the entries in the latent space are identical. Thus, we can use the residuals between the test data and their transformation by the generator to identify anomalies in the test data.

The Multivariate Time Series Anomaly Detection Using the combination of Temporal pattern and Feature pattern (MTAD-TF) [42] can be split into two main parts. The first part is called Temporal convolution component. It is based on a multiscale 1D convolution that allows to detect of temporal patterns. The second part is called Graph attention component. It allows one to learn the correlation between features and is based on a graph

attention network [79]. The combination of these two parts provides a prediction. The anomaly score is the squared error between the predicted and actual values.

Finally, **OmniAnomaly (OA)** [36] is a stochastic recurrent neural network for multivariate time series anomaly detection that learns robust multivariate time series representations with a stochastic variable connection and a planar normalizing flow that uses the reconstruction probabilities to determine anomalies. OmniAnomaly uses the Gated Recurrent Unit (GRU) to model the time dependencies of multivariate time series. The method also uses a VAE to learn a mapping of the input data W to a representation in a latent space. To model time dependencies in the latent space, OmniAnomaly uses a stochastic variable connection technique. As suggested by [33], conditional probability can evaluate the reconstruction. The anomaly score used is then the probability of reconstruction. A high score means that the input can be well reconstructed. If an observation follows normal time series patterns, it can be reconstructed with high confidence. On the other hand, the lower the score, the less well the observation can be reconstructed and the more likely it is to be anomalous.

2.10 Related Work

Anomaly detection is challenging, and many approaches have been taken in various applications. In past years, many classical unsupervised approaches have been developed [80-85], including Principal Component Analysis (PCA) [86], which finds a low-dimensional projection that captures most of the variance in the data. The anomaly score is the reconstruction error of this projection. It is a linear algebra technique that can automatically achieve dimension reduction. Lee et al. [87] proposed online over-sampling PCA, which makes use of online platforms for large-scale problems. By over-sampling the minority class of the target instance, their proposed algorithm allows them to determine the anomaly of the target instance.

One of the latest techniques for dimensionality reduction is Autoencoder [88], which is a popular approach for anomaly detection. Autoencoders consists of an encoder and decoder, which reconstruct data samples and use the reconstruction error as the anomaly score [89]. Zhou

et al. [90] proposed a deep autoencoder that combines robust PCA and deep autoencoders. It splits data into two parts: one part can be reconstructed by autoencoders; the other is the noise (outliers) in the data. Deep Autoencoding Gaussian Mixture Model (DAGMM) [31] jointly considers the Deep Autoencoder and Gaussian mixture Model to model the density distribution of multi-dimensional data.

Recently, Generative Adversarial Networks (GANs) [66] and LSTM-based approaches [62] have also shown promising performance for multivariate anomaly detection [91,92]. Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks (MAD-GAN) [37] proposed an unsupervised anomaly detection method based on generative adversarial networks (GAN) by considering complex dependencies among different time series variables. The LSTM-based Encoder-Decoder [27] models time series temporal dependency by means of LSTM networks and achieves better generalization capability than traditional methods. OmniAnomaly [36] is a stochastic recurrent neural network designed to avoid potential misguiding by uncertain instances, which uses stochastic variable connection and normalizing flow to get reconstruction probabilities to determine anomalies. Ryota et al. [93] introduced a convolutional neural network and environment-dependent anomaly detector to detect the object, its attributes, and actions in the image. An environment-specific model can identify unusual attributes likely to explain abnormal patterns. Hu et al. [94] proposed a time series anomaly detection technique using six meta-features. This technique is a One-class Support Vector Machine (OC-SVM) system designed to identify the abnormal states of a univariate or multivariate time series based on local dynamics. Recently, a novel computational approach, namely Local Recurrence Rate based Discord Search (LRRDS), was proposed to identify discords from multivariate time series. This approach reduces the dimensionality of a time series and can detect variable-length discords using the given time series as the normality reference [95].

U-Net [96] shares some of the design characteristics of the system architecture described in this paper. It is a fully convolutional neural network incorporating so-called skip channels

between encoding and decoding layers. This approach allows for integrating high and low level features in a way that prevents information loss and reduces the depth of sequential layers. However, U-net is limited in that the learning rate may be diminished in the middle layers of a high depth case. This means the system is at risk of ignoring layers with abstract features, thus limiting extraction of some of the complex features that could help image segmentation in medical images. Moreover, U-Net requires rather substantial training time because of a large number of hyper-parameters. By contrast, our system uses a more effective method consisting of the multi-scale MSCVAE model that has an attention-based ConvLSTM network. This method allows for capturing interesting features and key patterns to assist with anomaly detection in multivariate time series data.

Some Variational Autoencoders (VAEs) [63, 97-100] have taken a probabilistic approach, and autoencoders have been combined with Gaussian mixture modeling [101]. Bayer and Osendorfer [102] used Variational Inference to learn the underlying distribution of sequences and introduced recurrent stochastic networks. The core of these models is an RNN extended with a latent variable. Sölch et al. [103] used a Stochastic Recurrent Network (STORN) to detect robot anomalies using unimodal signals. Park et al. [38] presented the combination of LSTM-VAE using multimodal sensory signals, where LSTM is used to replace the feed-forward network in VAE. Pereira et al. [104] proposed applying a self-attention mechanism to VAE to improve the encoding-decoding process for energy data. Despite the intrinsic unsupervised setting, most of them may still be unable to detect anomalies effectively since most of the methods cannot capture temporal dependencies across different time steps in multivariate time series data.

2.11 Summary

In this chapter, the fundamental theories: Time series, Anomaly Detection, Variational Autoencoder, Long Short Term Memory, Attention mechanism, Moving Average, and evaluation metrics, as described above, will be used to create and develop the anomaly detection

framework for multivariate time series data. The methodology of using mentioned theories will be discussed in Chapter 3.

Chapter 3

Methodology

This chapter presents a Multi-scale convolutional variational autoencoder (MSCVAE) to detect anomalies in multivariate time series data. We first show how to generate multi-scale system attribute matrices. Then, All the components/layers of the proposed model are described in detail in this part.

3.1 Problem Statement

In this work, we focus on multivariate time series, given the historical data of n time series $X = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^{n \times T}$, of length T, and assume that there are anomalies in the data. We aim to detect anomalous events at certain time steps after T. We use only the normal dataset for training to characterize the various time series patterns under normal conditions. For the validation, we also use only the normal dataset. Both normal and abnormal data are used for testing.

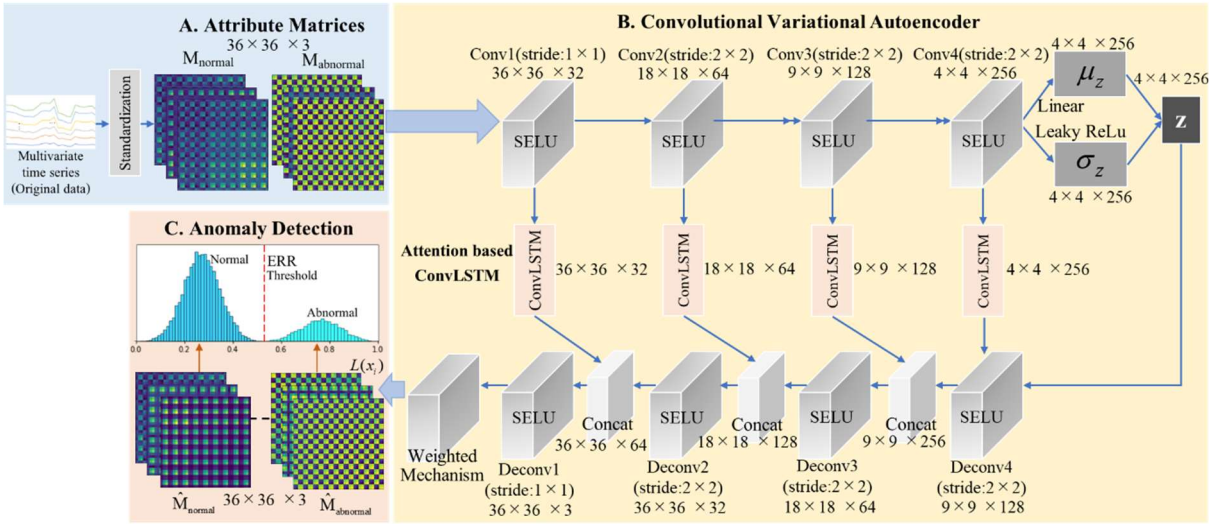


Fig. 3-1 Architecture of the MSCVAE framework

Table 3.1 Terminology and notation used in this thesis

Notations	Descriptions
X	Multivariate time series dataset
T	Length of time series
n	Dimension of time series
w	Size of sliding window
t	Time step
M^t	Attribute matrix
$p(z)$	Prior distribution
$p(z\|x)$	Probabilistic encoder
$p(\mathrm{x}\|z)$	Probabilistic decoder
$P^{t,0}$	Input of the first layer of the encoder
$N(0,I)$	A multivariate unit Gaussian distribution
z	Latent variable
$*$	Convolutional operation
$\circ$	Hadamard product
$\circledast$	Deconvolutional operation
$\oplus$	Concatenation operation
$f(\cdot)$	Activation function unit
$\hat{P}^{t,0}$	Output, reconstruction at t time step
$i^{t,l}$, $f^{t,l}$, $o^{t,l}$	3D tensor
$G^{t,l}$	Candidate memory
$C^{t,l}$	Cell state
$H^{t,l}$	Current hidden state

3.2 Methodology

3.2.1 Standardization

Data scaling is a recommended pre-processing step when working with many machine learning algorithms. Machine learning models learn a mapping from input variables to an output variable. The scale and distribution of the data drawn from the domain may be different for each variable. Input variables may have different units (e.g., feet, kilometers, and hours) that, in turn, may mean the variables have different scales.

In this thesis, we used standardization in which we scaled the data in order to have zero mean and unit variance. This can be thought of as subtracting the mean value or centering the data as shown in Eq. 3-1.

$$x = \frac{x_i - \mu}{\sigma} \tag{3-1}$$

where: x_i = sample point

μ = mean of the training samples

σ = standard deviation of the training samples

3.2.2 Generate Attribute Matrices

Given the importance of correlations between the different pairs of time series for characterizing the system state [105], we generate an $n \times n$ attribute matrix M^t utilizing the pair-wise inner product of multivariate time series within this segment. This is designed to illustrate the inter-correlations between different pairs of time series in a multivariate time series segment from time $t-w$ to t. We adopt the method for calculating the attribute matrix proposed in [39] to capture the similarity of shape and value scale correlations between two time series. Examples of attribute matrices are shown in Figure 3-1, part A. The pseudocode of the algorithm for generating the attribute matrix is introduced in Algorithm 1. For multivariate time series segment X^w, we define two time series, namely, $x_i = \left\{x_i^{t-w}, x_i^{t-w+1}, ..., x_i^t\right\}$, $x_j = \left\{x_j^{t-w}, x_j^{t-w+1}, ..., x_j^t\right\}$, and their correlation $m_{ij}^t \in M^t$ can be computed as follows:

$$m_{ij}^t = \frac{\sum_{\gamma=0}^{w} x_i^{t-\gamma} x_j^{t-\gamma}}{k} \tag{3-2}$$

where w is the length of sliding window, and k is a coefficient that is experiencedly set as w, which means the calculation of Eq. 3-2. Also, the interval between two segments is set to 10. To characterize latent features of multivariate time series in multi scales, we construct three channel attribute matrices with different length sliding windows (w =10, 20, 30) at each time step. To

further explain the attribute matrices generation process, the algorithm's pseudocode is shown in Table 3.2. An example of attribute matrices is shown in Figure. 3-2.

Table 3.2 Generating attribute matrices

Algorithm 1 Generating attribute matrices

Input:

X: Multivariate time series
T : Length of time series
w: Length of sliding window
n: Dimension of time series

Output:

m: Attribute matrix

1: **for** i in T **do**
2: for j in T **do**
3: $X_i = x[t-w+1], x[t-w+2], \ldots, x[t]$
4: $X_j = x[t-w+1], x[t-w+2], \ldots, x[t]$
5: $m[i][j] = \dfrac{X_i \times X_j}{w}$
6: return M

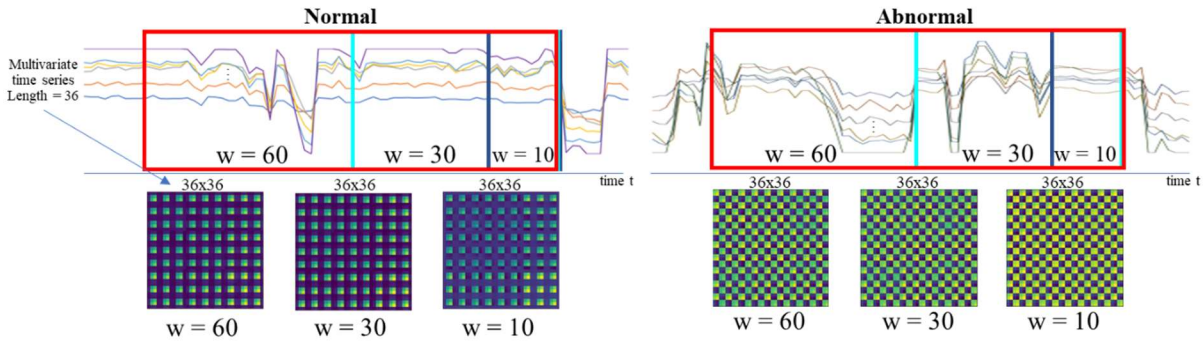


Fig. 3-2 The example of attribute matrices

3.2.3 Convolutional Encoder

We use convolution encoders [106] to filter the noise in the data and encode the spatial form of the system attribute matrices M. Four convolutional encoder layers are applied in our

model to extract the values of the attribute matrices in our framework. We call the input of the first layer $P^{t,0}$ at this time t for convenience. $P^{t,0}$ represents the input of the first layer and assumes that $P^{t,l-1} \in \mathbb{R}^{n_{l-1} \times n_{l-1} \times d_{l-1}}$ denotes the feature maps in $(l-1)-\text{th}$ layer. The output of $l-\text{th}$ layer is given by:

$$P^{t,l} = f(W_e^{\,l} * P^{t,l-1} + b_e^{\,l}) \tag{3-3}$$

where $*$ denotes the convolutional operation, $W_e^{\,l}$ is the filter kernel of layer l, $b_e^{\,l}$ is a term of bias, $P^{t,l}$ is the output of layer l, $P^{t,l-1}$ is the output of the $l-1$ layer (input of layer l), and $f(\cdot)$ is an activation function. Figure 3-1. part B illustrates the detailed encoding process of attribute matrices.

3.2.4 Variational Layer

The prior distribution over the latent variables, $p_\theta(z)$, is defined as an Isotropic Gaussian distribution, i.e. $p_\theta(z) = \text{Normal}(0, \text{I})$. The variational parameters of the approximate posterior $\tilde{q}_\phi(z|x)$, the mean μ_z and the standard deviation σ_z, are derived from the final encoder hidden state, Conv4, using two fully connected layers with Linear and Leaky Relu activations, respectively. Since $p(P^{t,l}|x)$ and $q_\phi(z|x)$ both have Normal distributions, the approximate posterior given a distribution around x, denoted $\tilde{q}_\phi(z|x)$, can be represented as a mixture of Gaussians. However, for computational convenience and following the approach of Park et al. [38] a single Gaussian is employed, i.e. $\tilde{q}_\phi(z|x) \approx q_\phi(z|P^{t,l})$. The latent variables are then obtained by sampling from the approximate posterior, $z \sim N(\mu_z, \sigma_z^2 \text{I})$, using the parametrization trick,

$$z = \mu_z + \sigma_z \odot \varepsilon \tag{3-4}$$

where $\varepsilon \sim N(0, I)$ is an auxiliary noise variable and $\odot$ represents an element-wise product.

3.2.5 Attention-based ConvLSTM

The spatial feature maps are extracted by the convolutional encoder, depending on the previous time steps. The traditional Convolutional LSTM performance might degrade as the length of the sequence increases. Xingjian et al. [107] introduced the LSTM model to the ConvLSTM to capture the temporal information in a video sequence. Here we have used an attention-based ConvLSTM to obtain the temporal characteristics which can be adaptive with different time steps. ConvLSTM consists of four convolutional layers of four ConvLSTM blocks. Details of ConvLSTM are shown in Figure 3-1. part B At time t, the results of $P^{t,l}$ calculated by Eq. 3-3. in layer l of the convolutional encoder are used as input to the $l-\text{th}$ layer of ConvLSTM. The other input is the time $t-1$ hidden state $H^{t-1,l}$ in hidden layer l. The ConvLSTM cell is formulated as follow:

$$
\begin{aligned}
i^{t,l} &= \sigma(W_{pi}^{l} * P^{t,l} + W_{hi}^{l} * H^{t-1,l} + W_{ci}^{l} \circ C^{t-1,l} + b_i^{l}) \\
f^{t,l} &= \sigma(W_{pf}^{l} * P^{t,l} + W_{hf}^{l} * H^{t-1,l} + W_{cf}^{l} \circ C^{t-1,l} + b_f^{l}) \\
o^{t,l} &= \sigma(W_{po}^{l} * P^{t,l} + W_{ho}^{l} * H^{t-1,l} + W_{co}^{l} \circ C^{t-1,l} + b_o^{l}) \\
G^{t,l} &= \tanh(W_{pc}^{l} * P^{t,l} + W_{hc}^{l} * H^{t-1,l} + b_c^{l}) \\
C^{t,l} &= f^{t,l} \circ C^{t-1,l} + i^{t,l} \circ G^{t,l} \\
H^{t,l} &= o^{t,l} \circ \tanh(\mathrm{C}^{t,l})
\end{aligned}
\tag{3-5}
$$

where σ is the sigmoid function, W_{pi}^{l} is the filter kernel of the input gate, W_{ci}^{l} is the filter kernel of the input gate process for the input of the hidden layer at the previous time step, and W_{hi}^{l} is the filter kernel of the cell state at the previous time step in the input gate. The input $P^{t,l}$, the cell state $C^{t,l}$, the hidden state $H^{t,l}$, the candidate memory $G^{t,l}$, and the gates $i^{t,l}$, $f^{t,l}$, $o^{t,l}$ are all 3D tensors. The symbol "$*$" denotes the convolutional operator, and "$\circ$" denotes the Hadamard product. Figure 3-1. part B illustrates the temporal modeling procedure.

We follow the idea of the temporal attention mechanism mentioned in [17], which is given by:

$$\hat{H}^{t,l} = \sum_{i \in (t-h,t)} \alpha^{i} H^{i,l} \tag{3-6}$$

$$\alpha^{i} = \frac{\exp(s^{t})}{\sum_{i \in (t-h,t)} \exp(s^{t})} \tag{3-7}$$

$$s^{t} = \frac{(H^{t,l})^{T}(H^{i,l})}{X} \tag{3-8}$$

where χ is the rescale factor (χ = 5). That is, we take the last hidden state $H^{t,l}$ as the group level context vector and measure the importance weights α^{i} of previous steps through a softmax function.

3.2.6 Convolutional Decoder

We can reconstruct the input data using the extracted attribute output produced by the decoder. For the convolutional decoder, we follow the procedure discussed in [39], which is formulated as follows:

$$\hat{P}^{t,l-1} = \begin{cases} f(\hat{W}_{d}^{t,l} \circledast \hat{H}^{t,l} + \hat{b}_{d}^{t,l}), & l = 4 \\ f(\hat{W}_{d}^{t,l} \circledast [\hat{H}^{t,l} \oplus \hat{P}^{t,l}] + \hat{b}_{d}^{t,l}), & l = 3,\ 2,\ 1 \end{cases} \tag{3-9}$$

where $\circledast$ denotes the deconvolutional operation, $\oplus$ is the concatenation operation, $f(\cdot)$ is the activation function unit (same as in the encoder). $\hat{W}_{d}^{t,l}$ and $\hat{b}_{d}^{t,l}$ are the filter kernel and bias parameter, respectively, of the $l-$ th deconvolutional layer. We reconstruct the attribute matrix of each layer by reversely decoding the deconvolution from layer l = 4 to layer l = 1. We then deconvolve the result $o^{t,4}$ of the ConvLSTM at the last layer l = 4 and reconstruct the previous layer matrix $\hat{P}^{t,3}$. After that, the deconvolution operation is performed as a concatenation to reconstruct the attribute matrix sequence of the $l-1-$ th layer. The final output $\hat{P}^{t,0} \in \mathbb{R}^{n \times n \times s}$

(with the same size as the input matrices) denotes the representations of reconstructed attribute matrices.

3.2.7 Weighted Mechanisms

The weighted mechanism aims to give different weights to temporal samples in each sliding window, and newer samples get higher weights, which means newer samples will be paid more attention. In contrast, the weights of older samples are lower, eliminating false alarms of the new sample under the excessive influence of historical samples. Linearly weighted moving average and exponentially weighted moving average are adopted to verify the effectiveness of the proposed weighted mechanism.

$$\mathrm{LWMA}_t = \frac{(\hat{P}_t^{t,l} * n) + (\hat{P}_{t-1}^{t,l} * (n\text{-}1)) + \ldots + (\hat{P}_{t-n+1}^{t,l} * (1))}{n(n+1)/2} \tag{3-10}$$

where LWMA_t is the value of the current period at time t, n is the number of periods, and $\hat{P}_t^{t,l}$ is the value of the series in the current period.

$$\mathrm{EWMA}_t = \alpha * \hat{P}^{t,l} + (1-\alpha) * \mathrm{EWMA}_{t-1} \tag{3-11}$$

where EWMA_t is moving average at time t, α is a degree of mixing parameter value between 0 and 1, and $\hat{P}^{t,l}$ is the value of the series in the current period. The coefficient α is set by experience. According to exponential weighting in Equation 3-11, coefficient α is set as 0.15.

3.2.8 Activation Function

Activation functions are mathematical equations used in neural networks for transforming the weighted sum of inputs from a node to its output. Activation functions can be either linear or non-linear. In this thesis, we use four activation functions defined as follows.

3.2.8.1 Linear

The linear activation function, also known as “no activation” or “identity function” (multiplied x1.0), is where the activation is proportional to the input.

The function does not do anything to the weighted sum of the input. It simply spits out the value it was given.

$$f(x) = x \tag{3-12}$$

The limitations of the Linear function are as follow:

- It is impossible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. Thus, essentially, a linear activation function turns the neural network into just one layer.

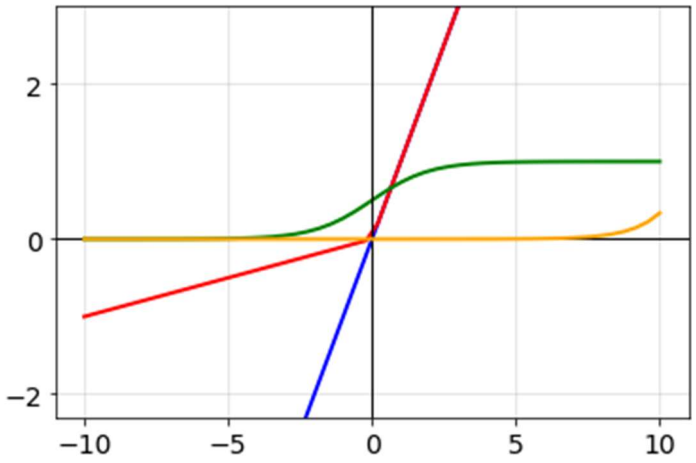


Figure 3-3 Activation Functions. Linear is used for deriving the expectation of the latent variables and the outputs; Sigmoid and Softmax are used internally in the attention based ConvLSTM; Leaky ReLu is used in the variance/diversity layers.

3.2.8.2 Leaky ReLu

Leaky ReLu is a recently developed activation function. It is designed to minimize sensitivity to the dying ReLU problem by having a small negative slope (in the neighborhood of 0.01), which is defined by:

$$f(x) = \max(0.01x, x) \tag{3-13}$$

The advantages of using Leaky ReLU as an activation function are as follows:

- Leaky ReLU is defined to address the problem of dying neurons/dead neurons.
- The problem of dying neuron/dead neuron is addressed by introducing a small slope having the negative values scaled by a enables their corresponding neurons to "stay alive".
- The function and its derivative are both monotonic.
- It allows negative value during backpropagation.
- It is efficient and easy for computation.
- Derivative of Leaky is 1 when $f(x) > 0$ and ranges between 0 and 1 when $f(x) < 0$

The limitations of the Leaky ReLu function are as follows:

- Leaky ReLU does not provide consistent predictions for negative input values

3.2.8.3 Sigmoid

This function takes any real value as input and outputs values in the range of 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below.

$$f(x) = \frac{1}{1+e^{-x}} \tag{3-14}$$

The advantages of using Sigmoid as an activation function are as follows:

- It is commonly used for models where we have to predict the probability as an output. Since the probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.

- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

The limitations of the Sigmoid function are as follows:

- Derivative of sigmoid function suffers "Vanishing gradient and Exploding gradient problem"
- Sigmoid function is not "zero-centric" This makes the gradient updates go too far in different directions. 0 < output < 1, and it makes optimization harder
- Slow convergence as its computationally heavy

3.2.8.4 Softmax

Softmax function is often described as a combination of multiple sigmoids. We know that sigmoid returns values between 0 and 1, which can be treated as probabilities of a data point belonging to a particular class. Thus, sigmoid is widely used for binary classification problems.

The Softmax function can be used for multiclass classification problems. This function returns the probability for a data point belonging to each individual class.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3-15}$$

3.2.9 Loss Function

The loss for a particular sequence X is given by:

$$\mathcal{L}(\phi, \theta; x^{(n)}) = \mathbb{E}_{z\sim\tilde{q}_\phi(z|x^{(n)})}[\log(p_\theta(x^{(n)}|z)] - D_{KL}(\tilde{q}_\phi(z|x^{(n)}) \,||\, p_\theta(z)) \tag{3-16}$$

Mean Square Error (MSE) is used as a reconstruction loss to optimize the parameter values in our model. MSE is given by the following:

$$L_{MSCVAE} = \sum_{t=1}^{k}\sum_{c=1}^{k}\left\|P^{t,0} - \hat{P}^{t,0}\right\|^2 \tag{3-17}$$

where $P^{t,0} \in \mathbb{R}^{n\times n}$. We employ the mini-batch stochastic gradient descent method and the Adam optimizer to minimize the above loss. After sufficient training epochs, the learned neural network parameters are utilized to infer the reconstructed attribute matrices of validation and test data. Finally, we perform anomaly detection and diagnosis based on the residual attribute matrices, elaborated in the next section.

3.2.10 Threshold Setting Strategy

VAEs map each group of attribute matrices to anomaly scores during the anomaly detection process. The threshold setting aims to give the best boundary to distinguish normal and abnormal samples, and then detected samples are labeled normal or abnormal by means of VAEs.

In a Receiver Operating Characteristic (ROC) curve [108], the true positive rate (TPR) is plotted in function of the false positive rate (FPR) for different cut-off points. Each point on the ROC curve indicates a pair of sensitivity and specificity corresponding to a specific decision threshold. Based on the ROC-based threshold setting strategy, the best threshold is always sought as the dot on the upper left corner or point (0,1). However, the ROC-based threshold setting strategy is insensitive to imbalanced datasets, leading to poor performance in anomaly detection. According to the false positive rate formula $\text{FPR} = \frac{FP}{FP+TN}$, when a large number of normal samples are wrongly judged as anomalies, FPR changes a little. Therefore, anomaly detection performance may be widely divergent for two threshold values that are close together. In this case, selecting an optimal threshold among different discrete points is not easy. Moreover, Eq. 3-12 is often adopted to calculate the distance between a point on the ROC curve and the point (0,1). The threshold corresponding to the minimum distance is selected as the best threshold.

However, TN is much larger than TP, which makes $\frac{FP^2}{FP^2+TN^2}$ much smaller than $\frac{FN^2}{FN^2+TP^2}$.

Therefore, when choosing the minimum distance, the ROC-based strategy focuses more on the former, only choosing the smaller FN as much as possible. Eventually, it leads to a low F1-Score.

$$\text{Distance} = \sqrt{(1-TPR)^2+FPR^2} \tag{3-18}$$

$$= \frac{FN^2}{FN^2+TP^2}+\frac{FP^2}{FP^2+TN^2} \tag{3-19}$$

We adopt a threshold setting strategy using a confusion matrix to avoid the abovementioned problems. The confusion matrix can effectively reflect the anomaly detection results, even for an imbalanced dataset. Therefore, we introduce a new error rate (ERR) defined as a function of TP, FP, FN, and TN, as shown in the formula below. The aim of the new threshold setting strategy is to minimize ERR, which means fewer samples are misjudged. As a result, the optimal threshold can be selected according to the minimum ERR.

$$\text{ERR} = \frac{FP}{FP+TP+TN} \tag{3-20}$$

As indicated above, Precision, Recall, F1-Score, and ERR are utilized as evaluation metrics for anomaly detection. Moreover, the geometric mean or G-mean [109] of sensitivity and specificity, is suitable for evaluating the quality of binary (two-class) classifications for a balanced as well as imbalanced dataset problem. Therefore, G-mean is used as another evaluation metric in the following experiments.

$$\text{G-mean} = \sqrt{\text{Recall}*\text{Specificity}}$$

$$= \sqrt{\frac{TP}{TP+FN}\times\frac{TN}{TN+FP}} \tag{3-21}$$

3.3 Summary

In this chapter, the MSCVAE proposed framework based on Variational Autoencoder is described. We explain how to generate the attribute matrices and show the example of attribute matrices of both normal and abnormal. Then, we encode the spatial information in feature matrices via a convolutional encoder and model the temporal pattern with attention based ConvLSTM. Finally, we reconstruct attribute matrices based on a convolutional decoder. We introduce in detail the strategy of threshold setting.

Chapter 4
Experiments and Discussion

This chapter presents the results of the thesis. The chapter begins with an explanation of the datasets used in this thesis. Afterward, the experimental setup and anomaly detection results are presented. This is followed by results from the ablation study, robustness evaluation, and threshold setting strategy comparison.

4.1 Datasets description

Four publicly available benchmark datasets were used in the experiments, namely, Satellite, Wafer (UCR), EEG, and Opt. Descriptions of these data sets are given below, and the details of the experiments are listed in Table 4.1.

Table 4.1 The detailed information of time series data sets

Datasets	#Training	#Validation	#Testing	Length	Anomaly rate
Satellite	3,000	1,000	2,435	36	31.6 %
Wafer	4,000	1,014	2,150	152	10.6 %
EEG	8,000	2,030	4,950	14	44.9 %
Opt	2,500	970	1,746	64	2.9 %

Satellite: This dataset was generated from data purchased from NASA by the Australian Centre for Remote Sensing. It consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image together with the classification associated with the central pixel in each neighborhood. The data set is used to predict the category of the image of the observed region of soil. Soil classes “red soil”, “grey soil”, “mixture class”, and “very damp grey soil” constitute the normal class. The class of anomalies consists of “cotton crop”, “damp grey soil”, and “soil with vegetation stubble”. This classification is based on multi-spectral values, consisting of 36 time series and 6435 instances.

Wafer: The Wafer dataset is related to semiconductor microelectronics fabrication, using data collected from various sensors during the processing of silicon wafers for semiconductor fabrication. Each time series in this dataset contains measurements recorded by one sensor in the course of processing one wafer by one tool. The dataset contains 152 attributes, and there is a large class imbalance between normal and anomaly.

EEG: This dataset is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. The eye state was detected via a camera during the EEG measurement and later added manually to the file after analyzing the video frames. This dataset consists of 14 EEG attribute values and one indicating the eye state.

Opt: The Opt dataset is used for preprocessing programs made available by NIST to extract normalized bitmaps of handwritten digits from a preprinted form, which contains 64 parameters. It is a character recognition dataset for integers 0-9. The 32X32 bitmaps are divided into non-overlapping blocks of 4X4 each, which generates an 8X8 input matrix. The instances of digits 1-9 are treated as inliers, whereas the instances of the digit 0 are treated as outliers.

4.2 Experimental setup

This experiment uses the open-source machine learning library Scikit-learn, and deep learning framework Torch, Keras, and TensorFlow to develop the baseline models and this framework. The computer configuration is Intel(R) Core(TM) i7-9750H CPU 2.60GHz 6-Core Processor with NVIDIA GeForce RTX 2060 GPU, 16G Memory.

Furthermore, seventeen algorithms were used to verify the superiority of the proposed framework, including classical anomaly detection algorithms and deep architecture models.

The classical anomaly detection algorithms include K-Nearest Neighbor (KNN), Local Outlier Factor (LOF), Isolation Forest (iForest), One-class Support Vector Machine (OCSVM), Gaussian Mixture Model (GMM), Principal Component Analysis (PCA), Angle-based Outlier

Detector (ABOD), Histogram-based Outlier Score (HBOS), Cluster-based Local Outlier Factor (CBLOF), Multiple-Objective Generative Adversarial Active Learning (MOGAAL).

The deep architecture models include Deep Autoencoding Gaussian Mixture Model (DAGMM), Generative Adversarial Network (GAN), Multivariate Anomaly Detection with GAN (MAD-GAN), OmniAnomaly, Autoencoder, Variational Autoencoder (VAEs), and Long Short-Term Memory Autoencoder (LSTM-AE). Table 4.2 presents the details of all the algorithms.

Table 4.2 Detailed structural comparison of the algorithms

Model	Kernel	Dense	CNN	LSTM	Dropout	Weighted Mechanism
KNN	×	×	×	×	×	×
LOF	×	×	×	×	×	×
iForest	×	×	×	×	×	×
OCSVM	○	×	×	×	×	×
GMM	×	×	×	×	×	×
PCA	×	×	×	×	×	×
ABOD	×	×	×	×	×	×
HBOS	×	×	×	×	×	×
CBLOF	×	×	×	×	×	×
MOGAAL	×	×	×	×	×	×
DAGMM	×	○	×	×	○	×
GAN	×	○	×	×	×	×
MAD-GAN	×	○	×	○	○	×
OmniAnomaly	×	○	×	×	×	×
Autoencoder	×	○	×	×	×	×
VAEs	×	○	×	×	×	×
LSTM-AE	×	○	×	○	○	×
MSCVAE	×	×	○	○	×	×
MSCVAE$_l$	×	×	○	○	×	○
MSCVAE$_e$	×	×	○	○	×	○

4.3 Anomaly detection results

4.3.1 Overall performance

All anomaly detection models in experiments were trained with the corresponding training subsets, consisting of normal samples. Then the models were verified using a validation method consisting of normal samples and testing subsets, including both normal and abnormal data. The model evaluation metrics, i.e., precision, recall, F1-Score, and G-mean, were in the range of 0 to 1. Higher precision, recall, F1-Score, G-mean, and lower ERR indicate better model performance. Table 4.3, Table 4.4, Table 4.5, and Table 4.6 show the confusion matrix elements and evaluation metrics of anomaly detection models on four datasets. The bold fonts in Table 4.3, Table 4.4, Table 4.5, and Table 4.6 indicate that the MSCVAE with exponential weighting is superior to the other models.

Table 4.3 Anomaly detection results of Satellite dataset

Model	Confusion matrix value				Performance evaluation				
	TP	FP	FN	TN	ERR	Precision	Recall	F1-Score	G-mean
KNN	45	75	68	51	0.4386	0.3750	0.3982	0.3863	0.4015
LOF	54	66	56	63	0.3607	0.4500	0.4909	0.4696	0.4896
iForest	51	69	58	61	0.3812	0.4250	0.4679	0.4454	0.4686
OCSVM	55	65	70	49	0.3846	0.4583	0.4400	0.4490	0.4349
GMM	72	48	73	46	0.2892	0.6000	0.4966	0.5434	0.4929
PCA	112	8	11	108	0.0351	0.9333	0.9106	0.9218	0.9207
ABOD	44	76	73	46	0.4578	0.3667	0.3761	0.3713	0.3766
HBOS	48	72	70	49	0.4260	0.4000	0.4068	0.4034	0.4059
CBLOF	51	69	62	57	0.3898	0.4250	0.4513	0.4378	0.4519
MOGAAL	52	68	78	41	0.4224	0.4333	0.4000	0.4160	0.3879
DAGMM	80	40	95	24	0.2778	0.6667	0.4571	0.5424	0.4140
GAN	85	37	78	39	0.2298	0.6967	0.5215	0.5965	0.5173
MAD-GAN	82	37	78	42	0.2298	0.6891	0.5125	0.5878	0.5220
OmniAnomaly	84	37	79	39	0.2313	0.6942	0.5153	0.5915	0.5142
Autoencoder	83	37	78	41	0.2298	0.6917	0.5155	0.5907	0.5206
VAEs	105	9	31	94	0.0433	0.9211	0.7721	0.8400	0.8394
LSTM-AE	105	11	70	53	0.0651	0.9052	0.6000	0.7216	0.7049
MSCVAE	122	4	6	107	0.0172	0.9683	0.9531	0.9606	0.9585
MSCVAE$_l$	124	3	6	106	0.0129	0.9764	0.9538	0.9650	0.9631
MSCVAE$_e$	125	2	4	108	**0.0085**	**0.9843**	**0.9690**	**0.9766**	**0.9754**

Table 4.4 Anomaly detection results of Wafer dataset

Model	Confusion matrix value				Performance evaluation				
	TP	FP	FN	TN	ERR	Precision	Recall	F1-Score	G-mean
KNN	58	48	37	67	0.2775	0.5472	0.6105	0.5771	0.5964
LOF	60	46	56	48	0.2987	0.5660	0.5172	0.5405	0.5139
iForest	54	52	43	61	0.3114	0.5094	0.5567	0.5320	0.5482
OCSVM	61	45	56	48	0.2922	0.5755	0.5214	0.5471	0.5187
GMM	60	46	45	59	0.2788	0.5660	0.5714	0.5687	0.5666
PCA	61	45	42	62	0.2679	0.5755	0.5922	0.5837	0.5858
ABOD	50	50	45	65	0.3030	0.5000	0.5263	0.5128	0.5454
HBOS	49	57	41	63	0.3373	0.7075	0.5034	0.5882	0.5346
CBLOF	33	73	62	42	0.4932	0.4623	0.5444	0.5000	0.3562
MOGAAL	45	61	50	54	0.3813	0.3113	0.3474	0.3284	0.4716
DAGMM	75	31	74	30	0.2279	0.4245	0.4737	0.4478	0.4975
GAN	120	7	15	68	0.1170	0.9449	0.8889	0.9160	0.8977
MAD-GAN	112	12	21	65	0.1864	0.9032	0.8421	0.8716	0.8431
OmniAnomaly	107	10	33	60	0.2575	0.9145	0.7643	0.8327	0.8094
Autoencoder	100	12	49	49	0.0745	0.8929	0.6711	0.7663	0.7342
VAEs	90	21	12	87	0.1061	0.8108	0.8824	0.8451	0.8431
LSTM-AE	86	9	64	51	0.0616	0.9053	0.5733	0.7020	0.6981
MSCVAE	142	2	0	66	0.0095	0.9861	**1.0000**	0.9930	0.9852
MSCVAE$_l$	144	1	0	65	**0.0048**	**0.9931**	**1.0000**	**0.9965**	**0.9924**
MSCVAE$_e$	142	1	1	66	0.0096	0.9930	0.9930	0.9930	0.9890

Table 4.5 Anomaly detection results of EEG dataset

Model	Confusion matrix value				Performance evaluation				
	TP	FP	FN	TN	ERR	Precision	Recall	F1-Score	G-mean
KNN	164	92	94	140	0.2323	0.6406	0.6357	0.6381	0.6193
LOF	152	104	123	111	0.2834	0.5938	0.5527	0.5725	0.5342
iForest	184	72	85	149	0.1778	0.7188	0.6840	0.7010	0.6791
OCSVM	163	93	96	138	0.2360	0.6367	0.6293	0.6330	0.6132
GMM	170	86	113	121	0.2281	0.6641	0.6007	0.6308	0.5926
PCA	189	67	76	158	0.1618	0.7383	0.7132	0.7255	0.7077
ABOD	160	96	111	123	0.2533	0.6250	0.5904	0.6072	0.5758
HBOS	178	78	77	157	0.1889	0.6953	0.6980	0.6967	0.6829
CBLOF	157	99	103	131	0.2558	0.6133	0.6038	0.6085	0.5865
MOGAAL	92	164	146	88	0.4767	0.3594	0.3866	0.3725	0.3674
DAGMM	194	62	173	61	0.1956	0.7578	0.5286	0.6228	0.5120
GAN	297	10	33	150	0.0219	0.9674	0.9000	0.9325	0.9186
MAD-GAN	297	26	27	140	0.0562	0.9195	0.9167	0.9181	0.8793
OmniAnomaly	296	14	38	142	0.0310	0.9548	0.8862	0.9193	0.8982
Autoencoder	225	34	142	89	0.0977	0.8687	0.6131	0.7188	0.6660
VAEs	259	16	23	192	0.0343	0.9418	0.9184	0.9300	0.9208
LSTM-AE	251	14	117	108	0.0375	0.9472	0.6821	0.7930	0.7770
MSCVAE	296	9	34	151	0.0197	0.9705	0.8970	0.9323	0.9201
MSCVAE$_l$	297	7	24	162	0.0150	0.9770	0.9252	0.9504	0.9418
MSCVAE$_e$	301	5	11	173	**0.0104**	**0.9837**	**0.9647**	**0.9741**	**0.9683**

Table 4.6 Anomaly detection results of Opt dataset

Model	Confusion matrix value				Performance evaluation				
	TP	FP	FN	TN	ERR	Precision	Recall	F1-Score	G-mean
KNN	82	23	13	52	0.1465	0.7810	0.8632	0.8200	0.7736
LOF	54	51	32	33	0.3696	0.5143	0.6279	0.5654	0.4967
iForest	78	28	24	40	0.1918	0.7358	0.7647	0.7500	0.6707
OCSVM	80	25	13	52	0.1592	0.7619	0.8602	0.8081	0.7622
GMM	82	23	14	51	0.1474	0.7810	0.8542	0.8159	0.7673
PCA	90	16	12	52	0.1013	0.8491	0.8824	0.8654	0.8214
ABOD	81	24	21	44	0.1611	0.7714	0.7941	0.7826	0.7168
HBOS	77	28	19	46	0.1854	0.7333	0.8021	0.7662	0.7061
CBLOF	82	23	13	52	0.1465	0.7810	0.8632	0.8200	0.7736
MOGAAL	51	54	58	7	0.4821	0.4857	0.4679	0.4766	0.2317
DAGMM	84	22	44	21	0.1732	0.7925	0.6563	0.7179	0.5661
GAN	97	1	6	66	0.0061	0.9898	0.9417	0.9652	0.9166
MAD-GAN	104	25	2	39	0.1488	0.8062	0.9811	0.8851	0.6688
OmniAnomaly	105	28	3	34	0.1677	0.7895	0.9722	0.8714	0.6073
Autoencoder	90	9	28	43	0.0634	0.9091	0.7627	0.8295	0.7942
VAEs	89	11	4	66	0.0663	0.8900	0.9570	0.9223	0.9057
LSTM-AE	92	3	35	40	0.0222	0.9684	0.7244	0.8288	0.8209
MSCVAE	99	1	4	66	0.0060	0.9900	0.9612	0.9754	0.9730
MSCVAE$_l$	104	5	4	57	0.0301	0.9541	0.9630	0.9585	0.9409
MSCVAE$_e$	103	1	0	66	**0.0059**	**0.9904**	**1.0000**	**0.9952**	**0.9925**

The comparison experiments show that 10 classical anomaly detection algorithms and 7 deep architecture models have been implemented. Generally speaking, the ability of classical anomaly detection algorithms is limited when facing modeling issues for multivariate time series, and this conclusion is supported by results for the two comprehensive indexes, F1 and G-mean. DAGMM reduces the dimension and extracts features by using neural networks, slightly improving compared to GMM. GAN achieves similar results to DAGMM on the Satellite dataset (shown in Table 4.3) but performs well on the other three datasets. MAD-GAN attempts to map data into the latent space and detect anomalies via discriminant results and

reconstruction errors generated from the mapping process. MAD-GAN achieves good results on the Opt dataset and gives slightly better results than OmniAnomaly. LSTM-AE performs better on low dimension datasets than high dimension datasets (shown in Tables 4.4 and 4.5). As an excellent sequence-to-sequence model, Autoencoder fails at temporal data and performs well on the Opt dataset as shown in Table 4.6. VAEs achieve better results than Autoencoder on four datasets, which suggests that a high dimension dataset is a challenge for the training of VAEs.

Compared to the algorithms above, three MSCVAE architecture-based algorithms are implemented. MSCVAE, MSCVAE$_l$, and MSCVAE$_e$ are respectively, MSCVAE model without weighting strategy, with linear weighting strategy and exponential weighting strategy. The comparison results demonstrate that MSCVAE with exponential weighting strategy is superior to the other algorithms discussed above, except in the Wafer dataset, which gives slightly better precision, recall, F1-Score, and G-mean values than MSCVAE with exponential weighting strategy. Furthermore, we can also come to the conclusion that no matter linear or exponential weighting strategies can improve the anomaly detection performance of multivariate time series effectively. In other words, MSCVAE is much better than baseline methods, as it can handle both inter-sensor correlations and temporal patterns of multivariate time series effectively.

The reasons for the superior performance of proposed framework can be summarized as follows: (1) As a deep learning model with temporal patterns, MSCVAE can achieve good performance with no need for supervised training. (2) VAEs with an attention-based ConvLSTM network framework can effectively identify anomalies. (3) the proposed framework can model both inter-sensor correlations and temporal patterns of multivariate time series effectively.

4.3.2 Ablation study

An extensive study illustrates the impact of different components on the model results, using two key modules of the proposed model: multi-scale attribute matrices and an attention-based ConvLSTM.

4.3.2.1 MSCVAE framework without weighted mechanism case (MSCVAE case)

Three variants of MSCVAE are considered in the evaluation:

- $MSCVAE_w$: MSCVAE framework without attention-based ConvLSTM.
- $CVAE_a$: MSCVAE framework without the multi-scale attribute matrices.
- $CVAE_w$: MSCVAE framework without both multi-scale attribute matrices and an attention-based ConvLSTM.

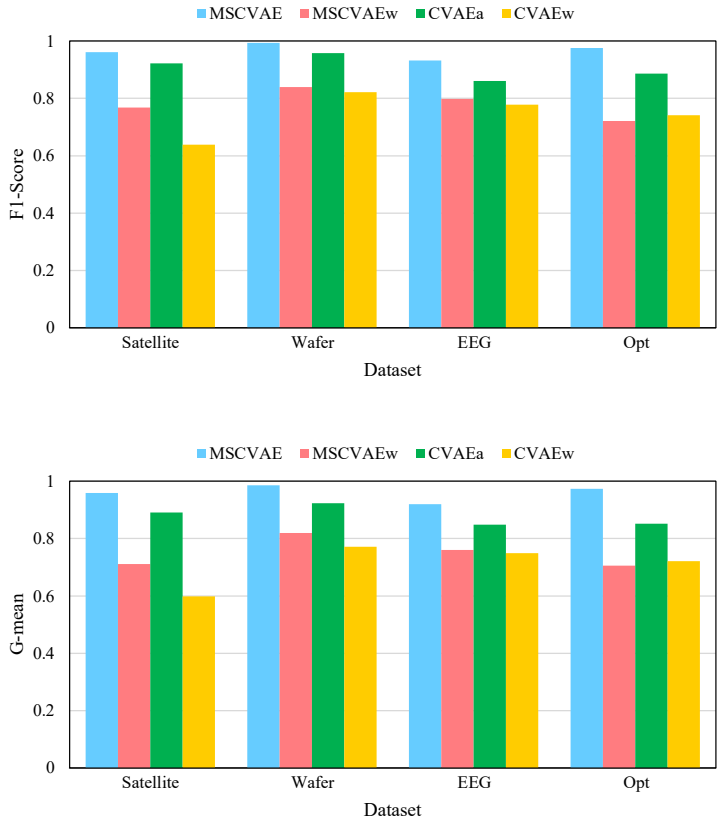


Fig. 4-1 F1-Score and G-mean comparison of all four datasets under three different competing models.

F1-Scores and G-means on four datasets are reported in Fig. 4-1. We observe that the proposed framework, MSCVAE (marked in blue), is obviously superior to the other three competing models on anomaly detection tasks, which indicates the importance of multi-scale attribute matrices and an attention-based ConvLSTM. However, $MSCVAE_w$ (marked in pink) and $CVAE_w$ (marked in yellow) methods can obtain approximately equal results with all four datasets. Besides, the results of $CVAE_a$ (marked in green) are much better than $MSCVAE_w$ and $CVAE_w$, which indicates the importance of an attention-based ConvLSTM. Furthermore, we see that under all conditions, both multi-scale attribute matrices and attention-based ConvLSTM can effectively improve the anomaly detection performance of multivariate time series.

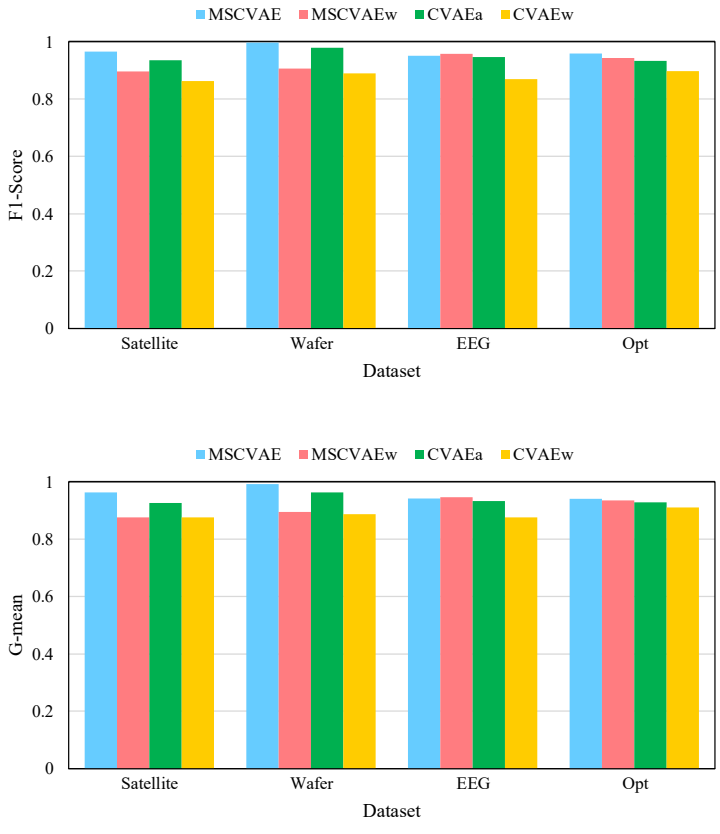


Fig. 4-2 F1-Score and G-mean comparison of all four datasets under three different competing models.

4.3.2.2 MSCVAE framework with linear weighted mechanism case (MSCVAE$_l$ case)

Three variants of MSCVAE$_l$ are considered in the evaluation:

- MSCVAE$_{l\text{-w}}$: MSCVAE$_l$ framework without attention-based ConvLSTM.
- CVAE$_{l\text{-a}}$: MSCVAE$_l$ framework without the multi-scale attribute matrices.
- CVAE$_{l\text{-w}}$: MSCVAE$_l$ framework without both multi-scale attribute matrices and an attention-based ConvLSTM.

F1-Scores and G-means on four datasets are reported in Fig. 4-2. We observe that the proposed framework, MSCVAE$_l$ (marked in blue), is superior to the other three competing models on anomaly detection tasks, except in the EEG dataset.

4.3.2.3 MSCVAE framework with exponentially weighted mechanism case (MSCVAE$_e$ case)

Three variants of MSCVAE$_e$ are considered in the evaluation:

- MSCVAE$_{e\text{-w}}$: MSCVAE$_e$ framework without attention-based ConvLSTM.
- CVAE$_{e\text{-a}}$: MSCVAE$_e$ framework without the multi-scale attribute matrices.
- CVAE$_{e\text{-w}}$: MSCVAE$_e$ framework without both multi-scale attribute matrices and an attention-based ConvLSTM.

F1-Scores and G-means on four datasets are reported in Fig. 4-3. We observe that the proposed framework, MSCVAE$_e$ (marked in blue), is superior to the other three competing models on anomaly detection tasks.

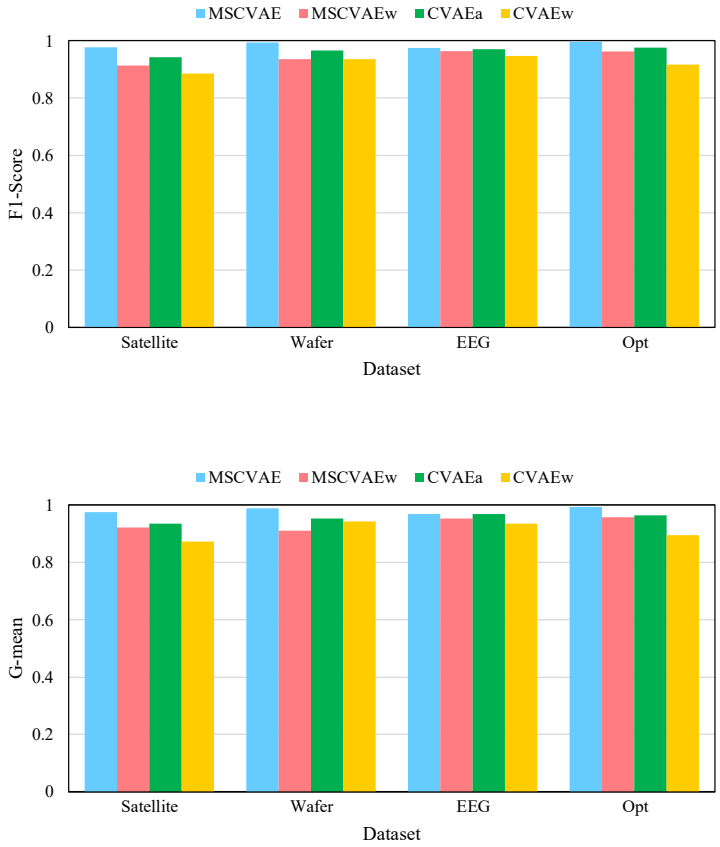
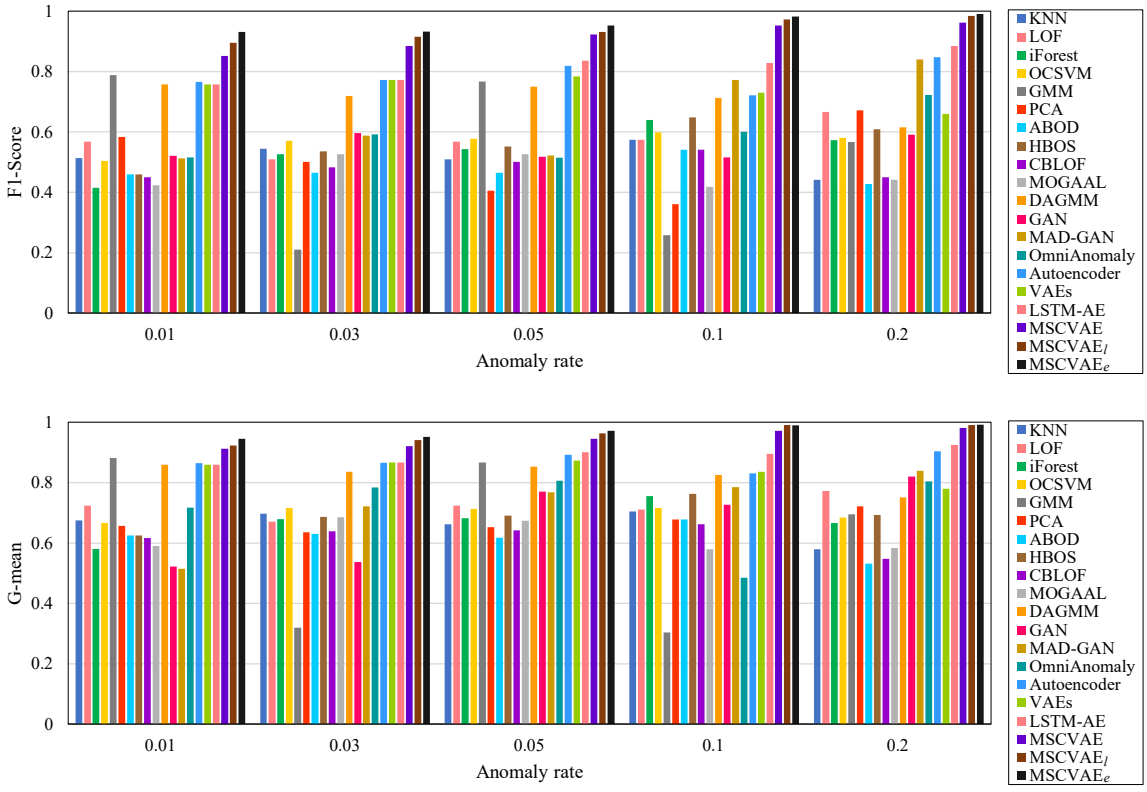


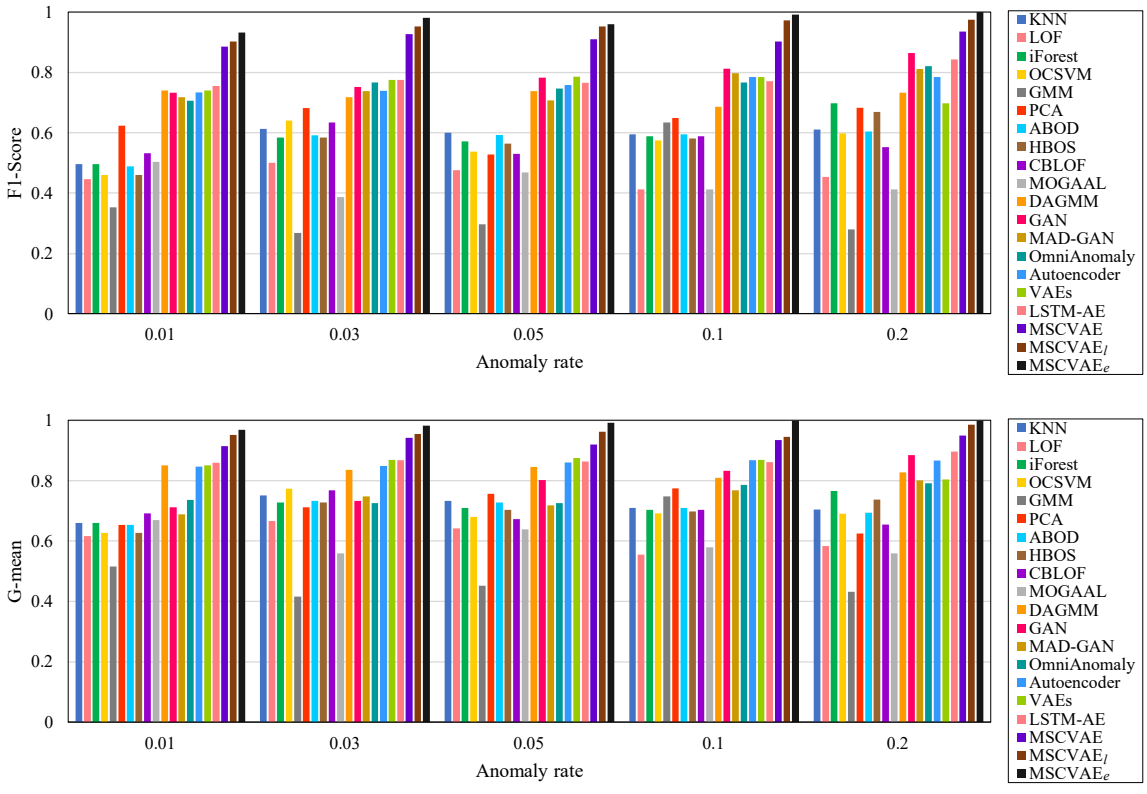
Fig. 4-3 F1-Score and G-mean comparison of all four datasets under three different competing models.

4.3.3 Robustness evaluation

Anomaly detection often suffers from the dataset imbalance problem, which means there are more normal samples than anomaly samples. To combat this problem, this thesis used the two criteria, F1-Score, and G-mean, with different rates of anomalies to evaluate the proposed model's robustness. Fig. 4-4 shows the chart of F1-Score and G-mean comparisons with different rates of anomalies. The comparison results indicate that the F1-Score and G-mean of most algorithms tend to increase as anomaly rates increase. The proposed framework's F1-Score and G-mean values are highest on all datasets under different anomaly rates. Therefore, we are justified in concluding that the proposed framework has good robustness even in the face of dataset imbalance problems.

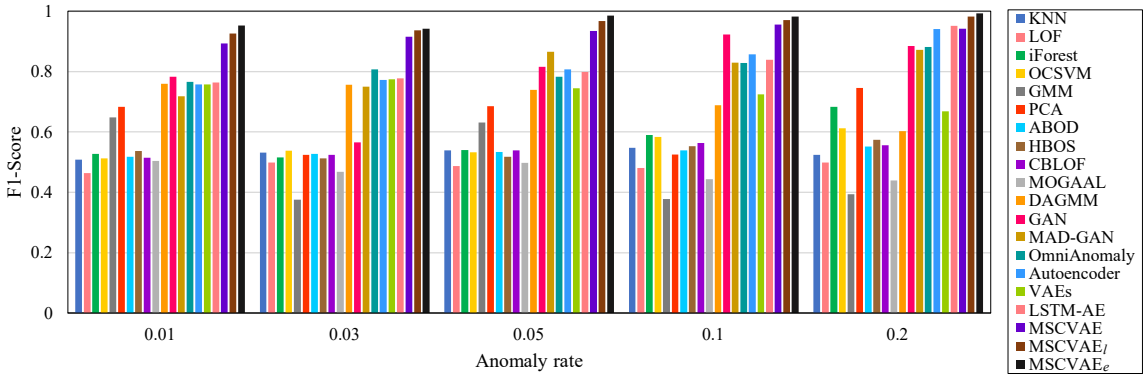


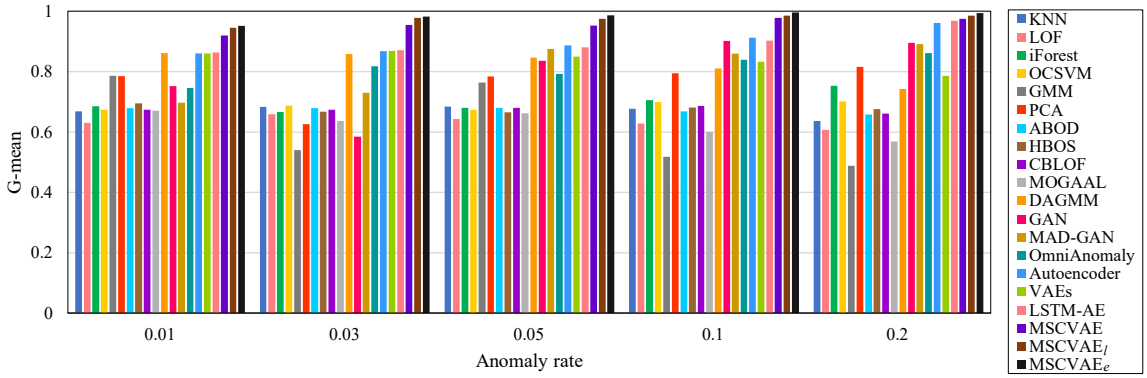
(a) Satellite dataset



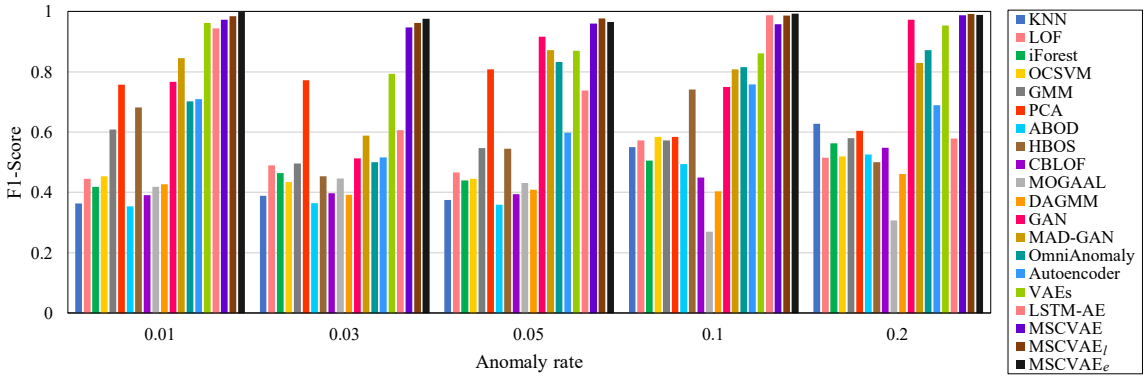
(b) Wafer dataset

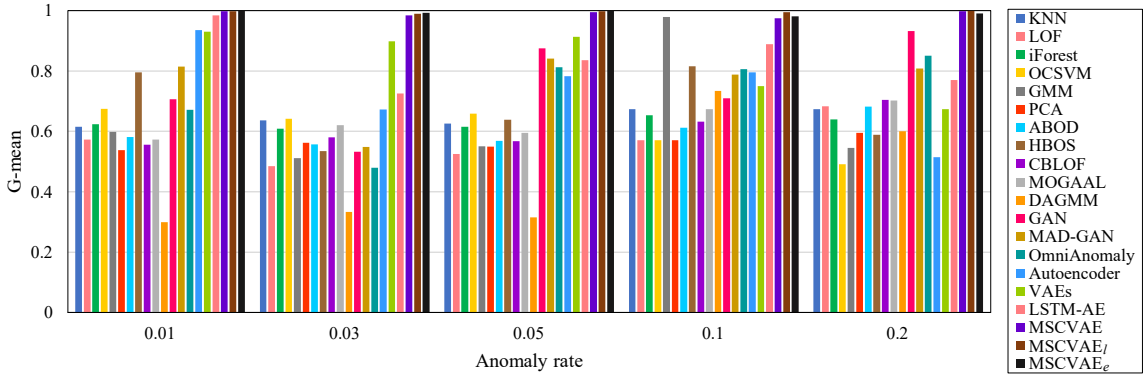
Fig. 4-4 F1-Score and G-mean comparison all four datasets under different anomaly rate.





(c) EEG dataset





(d) Opt dataset

Fig. 4-4 F1-Score and G-mean comparison all four datasets under different anomaly rate (continued)

4.3.4 Threshold setting strategy comparison

In order to verify the effectiveness of the proposed threshold setting strategy, comparing two threshold setting strategies via three comparative MSCVAE-based frameworks on four datasets, shown in Table 4.7. The comparison results demonstrate that the proposed ERR based threshold setting strategy is superior to ROC based strategy on all comparison models of Satellite dataset, EEG dataset, and Opt dataset, and the performance of the two strategies on Wafer dataset is quite the same. As stated, the proposed threshold setting strategy can effectively improve the model performance of multivariate time series anomaly detection. The bold fonts in Table 4.7 show us that new threshold setting strategy based anomaly detection can achieve better performance.

For further analysis, we conduct the two threshold setting strategies on MSCVAE based framework with exponential weighting strategy, and the detailed information is shown in Table 4.8, Table 4.9, Table 4.10, and Table 4.11. Table 4.8, Table 4.9, Table 4.10, and Table 4.11 show the top three thresholds on four datasets. The rank indicates the suitability of the threshold. Therefore, the highest rank is the best threshold. As explained in formulas 14 and 15, the threshold with the lower distance and the ERR will be a higher rank. Meanwhile, we also calculate TP, FP, FN, F1, and G-mean corresponding metrics under different thresholds. As stated, the proposed threshold setting strategy can effectively improve the model performance of multivariate time series anomaly detection.

Table 4.7 F1-Score and G-mean values of three MSCVAE based frameworks using two threshold setting strategies on four datasets.

Datasets (Anomaly rate)	Threshold setting strategy and anomaly detection model	F1-Score	G-mean
Satellite dataset (31.6 %)	MSCVAE + ERR based strategy	**0.9606**	**0.9585**
	MSCVAE + ROC based strategy	0.8632	0.8204
	MSCVAE$_l$ + ERR based strategy	**0.9650**	**0.9631**
	MSCVAE$_l$ + ROC based strategy	0.8294	0.7567
	MSCVAE$_e$ + ERR based strategy	**0.9766**	**0.9754**
	MSCVAE$_e$ + ROC based strategy	0.8750	0.8367
Wafer dataset (10.6 %)	MSCVAE + ERR based strategy	**0.9930**	**0.9852**
	MSCVAE + ROC based strategy	**0.9930**	**0.9852**
	MSCVAE$_l$ + ERR based strategy	**0.9965**	**0.9924**
	MSCVAE$_l$ + ROC based strategy	0.9952	0.9765
	MSCVAE$_e$ + ERR based strategy	**0.9930**	**0.9890**
	MSCVAE$_e$ + ROC based strategy	0.9930	0.9852
EEG dataset (44.9 %)	MSCVAE + ERR based strategy	**0.9323**	**0.9201**
	MSCVAE + ROC based strategy	0.9195	0.8824
	MSCVAE$_l$ + ERR based strategy	**0.9504**	**0.9418**
	MSCVAE$_l$ + ROC based strategy	0.9270	0.9168
	MSCVAE$_e$ + ERR based strategy	**0.9741**	**0.9683**
	MSCVAE$_e$ + ROC based strategy	0.9358	0.9043
Opt dataset (2.9 %)	MSCVAE + ERR based strategy	**0.9754**	**0.9730**
	MSCVAE + ROC based strategy	0.8851	0.7732
	MSCVAE$_l$ + ERR based strategy	**0.9585**	**0.9409**
	MSCVAE$_l$ + ROC based strategy	0.9223	0.9057
	MSCVAE$_e$ + ERR based strategy	**0.9952**	**0.9925**
	MSCVAE$_e$ + ROC based strategy	0.9952	0.9925

Table 4.8 Top three thresholds using strategies on Satellite dataset

Strategy	Rank	TP	FP	FN	TN	F1-Score	G-mean
ERR	1	125	2	4	108	0.9766	0.9754
	2	125	3	4	107	0.9728	0.9709
	3	124	4	6	105	0.9612	0.9586
ROC	1	126	29	0	84	0.8968	0.8622
	2	125	31	3	80	0.8803	0.8389
	3	124	37	2	76	0.8641	0.8136

Table 4.9 Top three thresholds using strategies on Wafer dataset

Strategy	Rank	TP	FP	FN	TN	F1-Score	G-mean
ERR	1	142	1	1	66	0.9930	0.9890
	2	142	2	3	63	0.9827	0.9743
	3	144	2	5	59	0.9763	0.9668
ROC	1	142	2	0	66	0.9930	0.9852
	2	140	5	3	62	0.9722	0.9518
	3	141	5	3	61	0.9724	0.9513

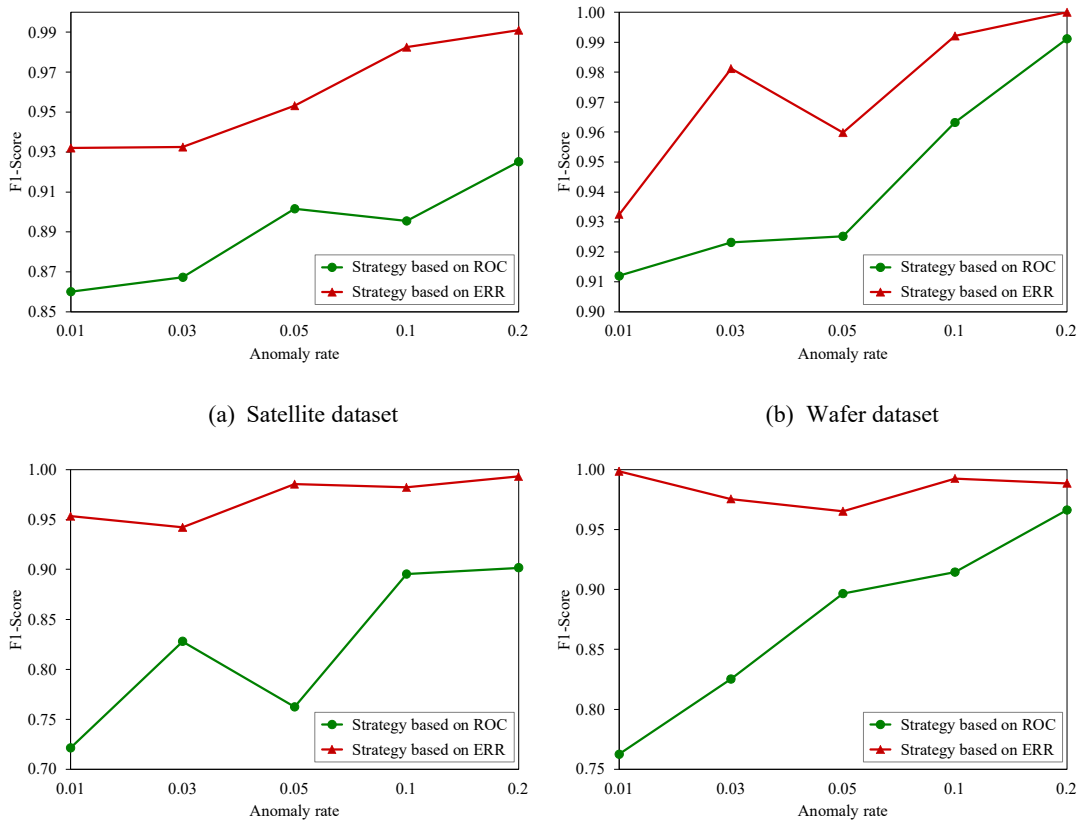
Table 4.10 Top three thresholds using strategies on EEG dataset

Strategy	Rank	TP	FP	FN	TN	F1-Score	G-mean
ERR	1	301	5	11	173	0.9741	0.9683
	2	301	6	11	172	0.9725	0.9655
	3	300	8	13	169	0.9662	0.9566
ROC	1	299	23	18	150	0.9358	0.9043
	2	300	37	7	144	0.9317	0.8817
	3	299	38	13	140	0.9214	0.8682

Table 4.11 Top three thresholds using strategies on Opt dataset

Strategy	Rank	TP	FP	FN	TN	F1-Score	G-mean
ERR	1	103	1	0	66	0.9952	0.9925
	2	102	1	1	66	0.9903	0.9877
	3	103	2	0	65	0.9904	0.9850
ROC	1	103	1	0	66	0.9952	0.9925
	2	103	2	0	65	0.9904	0.9850
	3	101	2	2	65	0.9806	0.9754

As stated, the ROC-based strategy pays more attention to FN and ignores FP, so it selects the best threshold with lower FN. In contrast, the ERR-based strategy focuses on FP, and its best selected threshold makes for similarly excellent FP performance. Consider the experimental results in Table 4.8. The first threshold of the ERR-based strategy has 2 FP and 4 FN, and the corresponding F1-Score and G-mean are 0.9766 and 0.9754. However, the first threshold of ROC-based strategy has 22 FP and 14 FN, and the corresponding F1-Score and G-mean are just 0.8750 and 0.8367. ROC-based strategy fails to achieve the best threshold because of its excessive attention to FN.



(a) Satellite dataset

(b) Wafer dataset

(c) EEG dataset

(d) Opt dataset

Fig. 4-5 F1-Score comparisons two threshold setting strategies on four datasets under different anomaly rate.

Further investigation was conducted to analyze the same proposed framework using the two threshold setting strategies under different anomaly rates, shown in Fig. 4-5. The ERR-based strategy achieves a better threshold than the ROC-based strategy under a low anomaly rate on four datasets. Besides, it is clear that the difference between the two strategies decreases as the anomaly rate increases. In short, the proposed threshold setting strategy is superior to the traditional ROC-based strategy and can achieve an appropriate threshold even when confronted with a dataset imbalance problem.

4.4 Discussion

In this part, we analyze and summarize the advantages of MSCVAE. The performance of MSCVAE is influenced by three components: the attribute matrices, an attention-based ConvLSTM, and a novel threshold setting strategy. The results of the experiments show that the proposed method on four standard datasets is better than the other seventeen evaluated algorithms. The reasons why MSCVAE is superior to the other algorithms of comparison can be summarized as follows:

First, when pre-processing data, an attribute matrices is calculated based inner-product for each time step, which contains the relationship between its own information and the information of a sub-sequence. That is why we can amplify features and reduce noise. The results of EEG dataset show low improvement compared to without attribute matrices, which can improve only 0.43%. Whereas, Opt dataset shows higher improvement compared to without attribute matrices, which can improve by 2.92%. Since the EEG dataset contains a little noise different from Opt dataset, which contains a lot of noise, therefore, attribute matrices can improve the data with higher noise.

Second, an attention based ConvLSTM is applied to select adaptively relevant hidden states (feature maps) across different time steps. That is why we can capture the temporal patterns of multivariate time series. The results of EEG dataset show low improvement compared to without attention-based ConvLSTM, which can improve only 1.09%. Whereas, Satellite dataset shows higher improvement compared to without attention based ConvLSTM,

which can improve by 6.34%. Since the EEG dataset has a low pattern appearance compared to Satellite dataset, therefore, Convlstm can not have the impact to improve.

Third, a new error rate (ERR) based threshold setting strategy is applied to optimize anomaly detection performance under an imbalance of normal and abnormal data. The ERR based strategy achieves a better threshold than the ROC-based strategy in Opt dataset, which can improve the F1-Score from 88.51% to 97.54%. Since Opt dataset has very much imbalanced: with anomalies of only 2.9% and a normal 97.1%, that is why Opt dataset can improve better than another dataset.

4.5 Summary

In this chapter, the overview of the proposed framework is thoroughly described and consists of three major processes: the pre-processing part, the convolutional variational autoencoder part, and the anomaly detection part. Experiments have been conducted on four datasets in order to verify the effectiveness of the proposed framework and the new ERR based threshold setting strategy.

Chapter 5
Conclusions

5.1 Conclusions

In this thesis, we proposed a novel MSCVAE framework to solve the anomaly detection problem for multivariate time series data. The framework employs multi scale (resolution) system attribute matrices which transform multivariate time series into multi scale attribute matrices. This approach allows for characterizing the state of the entire system in different time segments, and adopts the convolutional variational autoencoder to generate reconstructed attribute matrices, which makes the proposed framework more robust by taking advantage of VAEs. An attention-based Convolutional Long-Short Term Memory (ConvLSTM) network is used to capture the temporal patterns. The framework can model both inter-sensor correlations and temporal dependencies of multivariate time series. Finally, a new ERR-based threshold setting strategy is adopted, instead of a traditional ROC-based threshold setting strategy, to achieve better model performance. To verify the effectiveness of the proposed framework, experiments on four datasets were implemented. The results demonstrate that MSCVAE can outperform state-of-the-art baseline methods.

The work reported here justifies the following conclusions.

(1) Multi-scale attribute matrices provide an effective pre-processing method for characterizing system states at different time segments of multivariate time series with no need of prior knowledge.

(2) CNN structure is embedded in the encoder, and the decoder of the VAEs model is adopted to extract the characteristics of the time series. An attention-based Convolutional Long-Short Term Memory (ConvLSTM) network is used to capture the temporal patterns and reconstruct the attribute matrices, providing an effective unsupervised anomaly detection method. Combined with the proposed ERR-based threshold setting strategy, the MSCVAE based framework can achieve excellent performance.

(3) Experiments on four datasets indicate that the proposed framework outperforms competing models in detection accuracy and robustness under imbalanced datasets. An extensive experiment was also conducted to verify that the proposed threshold setting strategy can acquire an optimal threshold in the anomaly detection task, thus contributing to the superior anomaly detection performance of the proposed model.

5.2 Recommendations for future research

The present framework is able to be further extended to improve the overall performance of the proposed framework, some recommendations for future research are suggested as follows:

1. To increase the accuracy of anomaly detection by designing effective pre-processing, feature extraction should be developed to build a noise-insensitive framework for multivariate time series anomaly detection using U-Net which is based on an encoder-decoder neural network model.

2. To design the anomaly detection model that does not require a perfectly normal training set.

References

[1] P. J. Brockwell, and R. A. Davis, "Time series: theory and methods," *Springer Science & Business Media*, May, 13, 2009.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 15, 2009.

[3] L. Li, J. Yan, H. Wang, and Y. Jin, "Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 3, pp. 1177–1191, 2020.

[4] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proc. of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1939–1947, Aug. 2015.

[5] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2018.

[6] H. Izakian, and W. Pedrycz, "Anomaly detection and characterization in spatial time series data: A cluster-centric approach," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 6, pp. 1612–1624, 2014.

[7] R. Chalapathy, and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv:1901.03407*, 2019.

[8] Nilsonreport.com. Issue 1164. *Nilson Report*, 2019.

[9] M. Gupta, J. Gao, Y. Sun, and J. Han, "Community trend outlier detection using soft temporal pattern mining," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 692–708, 2012.

[10] E. J. Keogh, J. Lin, and A. W. Fu, "HOT SAX: Efficiently finding the most unusual time series subsequence," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*, pp. 226–233, 2005.

[11] S. Plakias, and Y. S. Boutalis, "Exploiting the generative adversarial framework for one-class multi-dimensional fault detection," *Neurocomputing*, vol. 332, pp. 396–405, 2019.

[12] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proc. of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, Jul. 2018.

[13] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *Proc. of IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1595–1598, Jul. 2018.

[14] S. Zavrak, and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, 2020.

[15] B. Yan, and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018.

[16] S. Du, T. Li, Y. Yang, and S. J. Horng, "Multivariate time series forecasting via attention-based encoder–decoder framework," *Neurocomputing*, vol. 388, pp. 269–279, 2020.

[17] M. Goldstein, and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, no. 4, pp. e0152173, 2016.

[18] J. Li, H. Izakian, W. Pedrycz, and I. Jamal, "Clustering-based anomaly detection in multivariate time series data," *Applied Soft Computing*, vol. 100, pp. 106919, 2021.

[19] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 2019.

[20] N. Jin, Y. Zeng, K. Yan, and Z. Ji, "Multivariate air quality forecasting with nested long short term memory neural network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8514-8522, 2021.

[21] H. Liang, L. Song, J. Wang, L. Guo, X. Li, and J. Liang, "Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series," *Neurocomputing*, vol. 423, pp. 444–462, 2021.

[22] A. Fernández, S. Garcia, M. Galar, C. R. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Berlin: Springer, 2018.

[23] A. Bayati, K. K. Nguyen, and M. Cheriet, "Multiple-Step-Ahead traffic prediction in high-speed networks," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2447–2450, 2018.

[24] P. Xiang, H. Zhou, H. Li, S. Song, W. Tan, J. Song, and L. Gu, "Hyperspectral anomaly detection by local joint subspace process and support vector machine," *International Journal of Remote Sensing*, vol. 41, no. 10, pp. 3798–3819, 2020.

[25] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proc. IJCNN*, vol. 3, pp. 1741–1745, 2003.

[26] T. F. Liu, M. K. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.

[27] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *IJCAI*, 2017.

[28] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] T. Yin, C. Liu, F. Ding, Z. Feng, B. Yuan, N. Zhang, "Graph-based stock correlation and prediction for high-frequency trading systems", *Pattern Recognition*, vol. 122, no. 108209, pp. 1-11, 2022.

[30] B. García-Martínez, A. Fernández-Caballero, R. Alcaraz, and A. Martínez-Rodrigo, "Assessment of dispersion patterns for negative stress detection from electroencephalographic signals," *Pattern Recognition*, vol. 119, no. 108094, pp. 1-9, 2021.

[31] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in Proc. *International conference on learning representations (ICLR)*, 2018.

[32] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," *MIT Press*, Cambridge, MA, USA, pp. 318-362, 1986.

[33] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng and J. Chen, “Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications,” In *Proceedings of world wide web conference,* pp. 187-196, 2018.

[34] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M.A. Zuluaga, “USAD: unsupervised anomaly detection on multivariate time series,” *In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* pp. 3395-3404, 2020.

[35] Z. Li, W. Chen, and D. Pei, “Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder,” *In 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC),* pp. 1-9, 2018.

[36] 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 Proc. *the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2828–2837, Jul. 2019.

[37] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S. K. Ng, “MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks,” in Proc. *International Conference on Artificial Neural Networks*, 2018.

[38] D. Park, Y. Hoshi, and C.C. Kemp, “A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544-1551, 2018.

[39] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, “A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data,” in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 1409–1406, 2019.

[40] M. Munir, S.A. Siddiqui, M.A. Chattha, A. Dengel, and S. Ahmed, “Fusead: unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models,” *Sensors*, vol. 19, no. 2451, pp. 1-15, 2019.

[41] N. Ding, H. Gao, H. Bu, H. Ma, and H. Si, “Multivariate-time-series-driven real-time anomaly detection based on bayesian network,” *Sensors*, vol. 18, no. 3367, pp. 1-13.

[42] Q. He, Y.J. Zheng, C.L. Zhang, and H.Y. Wang, "MTAD-TF: Multivariate time series anomaly detection using the combination of temporal pattern and feature pattern," *Complexity*, 2020.

[43] R.J. Hyndman, and G. Athanasopoulos, "*Forecasting: principles and practice*," OTexts, 2018.

[44] D.M. Hawkins, *Identification of outliers,* vol. 11, London: Chapman and Hall, 1980.

[45] 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.

[46] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognition*, vol. 74, pp.406-421, 2018.

[47] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," 2016.

[48] J. Patterson, and A. Gibson, "*Deep learning: A practitioner's approach*," O'Reilly Media, Inc, 2017.

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

[50] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q.V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," pp. 2978-2988, 2019.

[51] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PloS one*, 13(3), p.e0194889, 2018.

[52] L. Breiman, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical science*, vol. 16, no. 3, pp.199-231, 2001.

[53] K. Gurney, "An introduction to neural networks," CRC press, 1997.

[54] A. Graves, "Supervised sequence labelling," In Supervised sequence labelling with recurrent neural networks, pp. 15-35, Springer, 2012.

[55] A. Karpathy, "Convolutional neural networks for visual recognition," cs231n. github. io. 2018.

[56] R.A. Dunne, and N.A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," In *Proc. 8th Aust. Conf. on the Neural Networks*, Vol. 181, pp. 185, 1997.

[57] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp.219-269, 1995.

[58] Z.C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," arXiv preprint arXiv:1506.00019, 2015.

[59] C. Olah, "Understanding lstm networks," http: //colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[60] P.J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp.1550-1560, 1990.

[61] D. Britz, "Recurrent neural networks tutorial, part 3–backpropagation through time and vanishing gradients," http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/, 2015.

[62] S. Hochreiter, and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp.1735-1780, 1997.

[63] D.P. Kingma, and M. Welling, "Auto-encoding variational bayes," Proc. *Int. Conf. Learn. Represent.*, 2014.

[64] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp.183-233, 1999.

[65] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp.1798-1828, 2013.

[66] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, 27, 2014.

[67] C. Cremer, X. Li, and D. Duvenaud, "Inference suboptimality in variational autoencoders," In *International Conference on Machine Learning,* pp. 1078-1086, 2018.

[68] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp.359-366, 1989.

[69] A. Alemi, B. Poole, I. Fischer, J. Dillon, R.A. Saurous, and K. Murphy, "Fixing a broken ELBO," In *International Conference on Machine Learning,* pp. 159-168, 2018.

[70] R. Hübner, M. Steinhauser, and C. Lehle, "A dual-stage two-phase model of selective attention," *Psychological review*, vol. 117, no. 3, pp.759, 2010.

[71] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

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

[73] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156-3164, 2017.

[74] J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, 2015.

[75] E. Parisotto, D. Singh Chaplot, J. Zhang, and R. Salakhutdinov, "Global pose estimation with an attention-based recurrent network." In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops,* pp. 237-246, 2018.

[76] A.M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[77] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," In *International conference on machine learning,* pp. 2048-2057, 2015.

[78] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using vae-lstm hybrid model," In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4322-4326, 2020.

[79] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[80] B. Min, J. Yoo, S. Kim, and D. Shin, “Network Anomaly Detection Using Memory-Augmented Deep Autoencoder,” *IEEE Access*, vol. 9, pp. 104695–104706, 2020.

[81] S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, and K. Han, “Enhanced Network Anomaly Detection Based on Deep Neural Networks,” *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[82] A. Lavin, and S. Ahmad, “Evaluating real-time anomaly detection algorithms-the Numenta anomaly benchmark,” in *Proc. of the 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 38–44, Dec. 2015.

[83] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in *Proc. of international conference on artificial neural networks*, pp. 52–59, Jun. 2011.

[84] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, “Real-time big data processing for anomaly detection: A survey,” *International Journal of Information Management*, vol. 45, pp. 289–307, 2019.

[85] J. Fan, Q. Zhang, J. Zhu, M. Zhang, Z. Yang, and H. Cao, “Robust deep auto-encoding Gaussian process regression for unsupervised anomaly detection,” *Neurocomputing*, vol. 376, pp. 180–190, 2020.

[86] M. L. Shyu, S. C. Chen, K. Sarinnapakorn, and L. Chang, “A novel anomaly detection scheme based on principal component classifier,” MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2003.

[87] Y. J. Lee, Y. R. Yeh, and Y. C. F. Wang, “Anomaly detection via online oversampling principal component analysis,” *IEEE transactions on knowledge and data engineering*, vol. 25, no. 7, pp. 1460-1470, 2012.

[88] C. C. Aggarwal, “Outlier analysis,” Data mining: the textbook, Springer, 2015.

[89] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, “Anomaly detection using autoencoders in high performance computing systems,” in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 9428–9433, Jul. 2019.

[90] C. Zhou, and R. C. Paffenroth, “Anomaly detection with robust deep autoencoders,” in Proc. *the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 665–674, Aug. 2017.

[91] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *Proc. International conference on learning representations (ICLR)*, 2015.

[92] F. Karim, S. Majumdar, H. Darabi, and S. Chen, “Lstm fully convolutional networks for time series classification,” *IEEE Access*, vol. 6, pp. 1662–1669, 2017.

[93] R. Hinami, T. Mei, and S.I. Satoh, “Joint detection and recounting of abnormal events by learning deep generic knowledge,” In *Proc of the IEEE International Conference on Computer Vision*, pp. 3619-3627, 2017.

[94] M. Hu, Z. Ji, K. Yan, Y. Guo, X. Feng, J. Gong, X. Zhao, and L. Dong, “Detecting anomalies in time series data via a meta-feature based approach,” *IEEE Access*, vol. 6, pp. 27760-27776, 2018.

[95] M. Hu, X. Feng, Z. Ji, K. Yan, and S. Zhou, “A novel computational approach for discord search with local recurrence rates in multivariate time series,” Information Sciences, vol. 477, pp. 220-233, 2019.

[96] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention, pp. 234-241, Oct. 2015.

[97] J. Sun, X. Wang, N. Xiong, and J. Shao, “Learning sparse representation with variational auto-encoder for anomaly detection,” *IEEE Access*, vol. 6, pp. 33353–33361, 2018.

[98] X. Wang, Y. Du, S. Lin, P. Cui, Y. Shen, and Y. Yang, “adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection,” *Knowledge-Based Systems*, vol. 190, pp. 105187, 2020.

[99] R. Yao, C. Liu, L. Zhang, and P. Peng, “Unsupervised anomaly detection using variational auto-encoder based feature extraction,” in *Proc. 2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–7, Jun. 2019.

[100] J. An, and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[101] N. Li, and F. Chang, "Video anomaly detection and localization via multivariate gaussian fully convolution adversarial autoencoder," Neurocomputing, vol. 369, pp. 92–105, 2019, DOI: 10.1016/j.neucom.2019.08.044.

[102] J. Bayer, and C. Osendorfer, "Learning stochastic recurrent networks," Proc. *NIPS Workshop Advances Variational Inf*, 2014.

[103] M. Sölch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational inference for on-line anomaly detection in high-dimensional time series," Proc *ICML*, 2016.

[104] J. Pereira, and M. Silveira, "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention," In *IEEE international conference on machine learning and applications (ICMLA)*, pp. 1275-1282, 2018.

[105] D. Hallac, S. Vare, S. Boyd, and J. Leskovec, "Toeplitz inverse covariance-based clustering of multivariate time series data," in *Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 215–223, Aug. 2017.

[106] J. Long, E. Shelhamer, and T. Darrell, ''Fully convolutional networks for semantic segmentation,'' in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[107] S.H.I. Xingjian, Z. Chen, H. Wang, D.Y. Yeung, W.K. Wong, and W.C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," In *Advances in neural information processing systems*, pp. 802-810, 2015.

[108] K. Hajian-Tilaki, "Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation," *Caspian journal of internal medicine*, vol. 4, no. 2, pp. 627–635, 2013.

[109] X. Hua, Y. Cheng, H. Wang, Y. Qin, and Y. Li, "Geometric means and medians with applications to target detection," *IET Signal Processing*, vol. 11, no. 6, pp. 711-720, 2017.

Appendix

A. Publications and Presentations from the Present Research Work

Journal

1. U. Yokkampon, A. Mowshowitz, S. Chumkamon, and E. Hayashi, "Robust Unsupervised Anomaly Detection with Variational Autoencoder in Multivariate Time Series Data," *IEEE Access*, vol. 10, pp. 57835-57849, June 2022.
2. U. Yokkampon, A. Mowshowitz, S. Chumkamon, and E. Hayashi, "Autoencoder with Gramian Angular Summation Field for Anomaly Detection in Multivariate Time Series Data," *Journal of Advances in Artificial Life Robotics*, vol. 2, no. 4, pp. 423-427, March 2022.
3. U. Yokkampon, S. Chumkamon, A. Mowshowitz, R. Fujisawa, and E. Hayashi, "Anomaly Detection Using Support Vector Machines for Time Series Data," *Journal of Robotics, Networking and Artificial Life*, vol. 8, no. 1, pp. 41-46, 2021.
4. U. Yokkampon, S. Chumkamon, A. Mowshowitz, and E. Hayashi, "Autoencoder with Spiking in Frequency Domain for Anomaly Detection of Uncertainty Event," *Journal of Robotics, Networking and Artificial Life*, vol. 6, no. 4, pp. 231-234, 2020.
5. S. Chumkamon, K. Kawamoto, U. Yokkampon, and E. Hayashi, "Robot Motion and Grasping for Blindfold Handover," *Journal of Advances in Artificial Life Robotics*, vol. 1, no. 1, pp. 1-5, 2020.

Conference

1. S. Chumkamon, T. Tsuji, P. Gamolped, C. Piyavichyanon, U. Yokkampon, A. Mowshowitz, and E. Hayashi, "Autonomous Robotics Packaging Ready Meal in Conveyor Production Line," In *Proceedings of International Conference on Artificial Life & Robotics (ICAROB2022)*, pp. 584-588, January 20-23, 2022.
2. U. Yokkampon, A. Mowshowitz, S. Chumkamon, and E. Hayashi, "Anomaly Detection using Autoencoder with Gramian Angular Summation Field in Multivariate Time Series Data," In *Proceedings of International Conference on Artificial Life & Robotics (ICAROB2022)*, pp. 579-583, January 20-23, 2022.
3. S. Chumkamon, U. Yokkampon, E. Hayashi, and R. Fujisawa, "Robot Motion Generation by Hand Demonstration," In *Proceedings of International Conference on Artificial Life & Robotics (ICAROB2021)*, pp. 768-771, January 21-24, 2021.
4. U. Yokkampon, S. Chumkamon, A. Mowshowitz, and E. Hayashi, "Anomaly Detection in Time Series Data Using Support Vector Machines," In *Proceedings of International Conference on Artificial Life & Robotics (ICAROB2021)*, pp. 581-587, January 21-24, 2021.

5. U. Yokkampon, S. Chumkamon, A. Mowshowitz, R. Fujisawa, and E. Hayashi, “Improved Variational Autoencoder Anomaly Detection in Time Series Data,” In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 82-87, October 11-14, 2020.

6. U. Yokkampon, S. Chumkamon, A. Mowshowitz, and E. Hayashi, “Anomaly Detection using Variational Autoencoder with Spectrum Analysis for Time Series Data,” In *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 1-6, August 26-29, 2020.

7. S. Chumkamon, K. Kawamoto, J. Inthiam, U. Yokkampon, E. Hayashi, “Robot Motion and Grasping for Blindfold Handover,” In *Proceedings of International Conference on Artificial Life & Robotics (ICAROB2020)*, pp. 255-258, January 13-16, 2020.

8. U. Yokkampon, S. Chumkamon, A. Mowshowitz, and E. Hayashi, “Autoencoder with Spiking in Frequency Domain for Anomaly Detection of Uncertainty Event,” *In Proceedings of International Conference on Artificial Life & Robotics (ICAROB2020)*, pp. 245-248, January 13-16, 2020.

List of Figures

Page

List of Tables

Page

Acknowledgements

It is my pleasure to acknowledge the roles of several individuals who were instrumental for completion of my Ph.D. research.

First of all, I would like to express my sincere gratitude to my advisor, Professor Dr. Eiji HAYASHI, who has been a tremendous mentor and support for me. I deeply appreciate how you have been continuously encouraging all research and guiding me in the last three years to allow me to grow as a researcher.

Besides my advisor, I am thankful to Professor Dr. Abbe Mowshowitz from The City College of New York for agreeing to be a reviewer of my all research papers. Also, I always appreciate your advisement and warm encouragement.

I would also like to thank my committee members, Professor Dr. Masanobu KOGA, Professor Dr. Eiji MIYANO, Associate Professor Dr. Hiroshi OHTAKE for serving as my committee members. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

I am grateful to a researcher in my laboratory, Dr. Sakmongkon Chumkamon who helped me out with my doubts and always been ready to discuss over the past three years.

I am also extremely grateful to my fellow laboratory members that I had the opportunity to work with during the course of my Ph.D., especially Tsuji, Tomokawa, Tominaga, Noboru, and Kota for their help both inside and out of the laboratory.

I sincerely acknowledge the support from the Rotary Yoneyama Memorial Foundation scholarship and 100th Anniversary Memorial scholarship. Your generosity allows me to take my goals and dreams a reality and this scholarship has afforded me the opportunity to continue my educational pursuits.

Last but not least, I would like to thank my family for constantly being there for me and supporting me throughout my Ph.D.; without you none of this would indeed be possible.

I really could not have been able to complete my PhD without the support of all the aforementioned people. Thank you all so much.

UMAPORN YOKKAMPON