MSBD 5002 - Knowledge Discovery and Data Mining - Fall Semester (2021-2022)
Group Project – Multi-dataset Time Series Anomaly Detection (Group 24)

CHEN Congjian	LIU Yi	LIU Mengchen
20787824	20784561	20796095
cchenci@connect.ust.hk	yliujt@connect.ust.hk	mliubu@connect.ust.hk

I. Abstract

This is a project from KDD Cup 2021, one of the most famous data mining competitions. The major goal of this project is to detect and identify unnormal data points on time series data.

II. Introduction

Time series data is composed of a sequence of values over time, which exists in many applications, such as stock prices and websites. However, some values may deviate so much from other normal observations, and these anomalous values are called anomalies. Anomalous data can indicate critical incidents, such as technical incidents. Anomaly detection on time series data is to identify these anomalous data points, where machine learning technique is progressively being utilized to detect anomalies automatically.
In this project, we need to discover the location of the anomaly. For each time series data, it contains two sections, the first section is training data and is completely free of anomalies, and the second section is test data set that contains exactly one anomaly. In this project, we need to find the location of this anomaly.

III. Objective

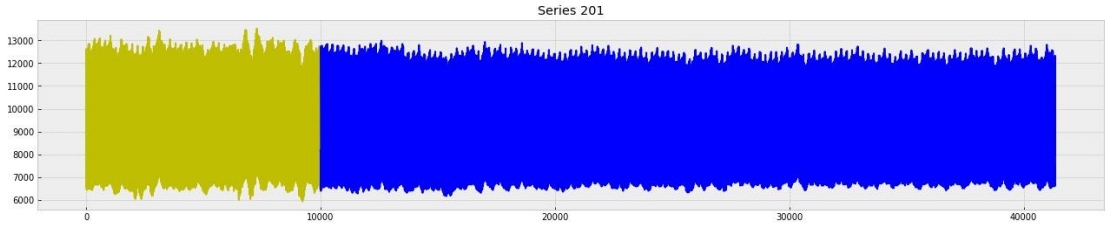
We have 4 objectives throughout the project, including:

- To acquire a better understanding of data mining techniques.
- To familiar how to complete a data mining and analytic project as a team
- To learn the procedures of data pre-processing of time series data
- To finish our task and detect the anomaly of time series data by using Matrix Profile model and LSTM model

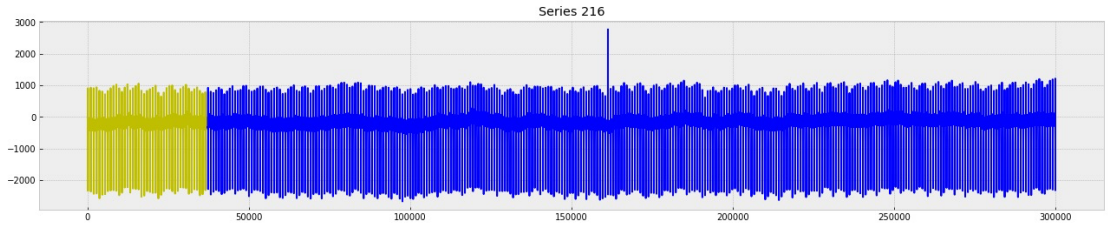
IV. Data Engineering

A. Series Overview

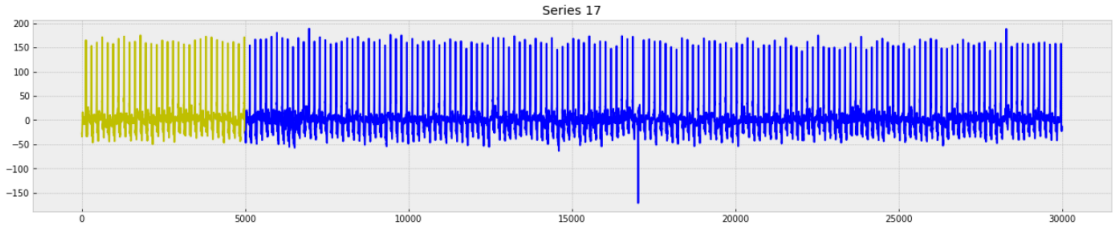
To have a rough understanding of the data, we firstly conduct a data feature detection for the whole data set, sample some of the dataset from the whole 250 dataset randomly, plot the time series of the data and observe the characteristics of the dataset.



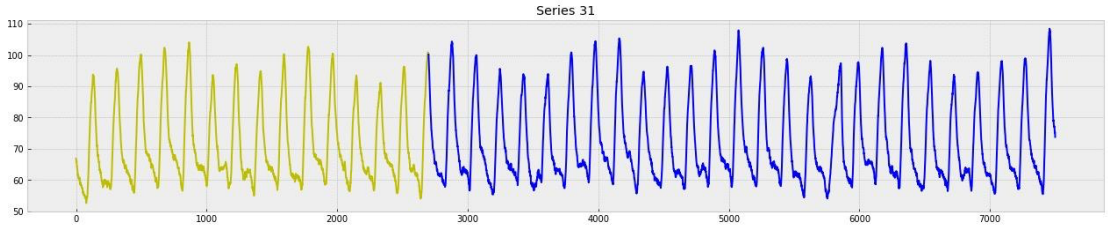
Stationary series without sudden maximum/minimum



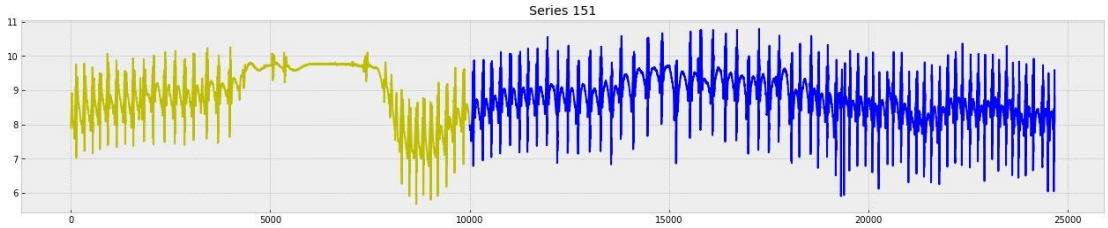
Stationary series with a sudden maximum/minimum



Stationary series with a sudden zero value



Non-stationary series with some seasonal features



Weird training set

Figure 1: Different series

From figure 1, we can see that time series in different data sets have different characteristics, which requires us do some further preprocessing on data.

B. Exponential Smoothing

After browsing through some datasets, we conduct ADF test, also known as unit root test, on all datasets. This test is to judge whether there is a unit root in the sequence. If

the sequence is stationary, there is no unit root. Otherwise, there will be unit roots. If and only if the p-value of the test is less than 5%, there will be no unit roots and the sequence is stationary.

Next, we need to perform some operations on all the non-stationary datasets to reduce the calculation workload later. The exponential smoothing method, which uses the exponential window function to assign weight decreasing over time, is much used under the situation. The formula is below:

$$\hat{y}_t = \alpha \cdot y_t + (1-\alpha) \cdot \hat{y}_{t-1}$$

Here the model value is a weighted average between the current true value and the previous model values. The α weight is called a smoothing factor. It defines how quickly we will "forget" the last available true observation. The smaller α is, the more influence the previous observations have and the smoother the series is.

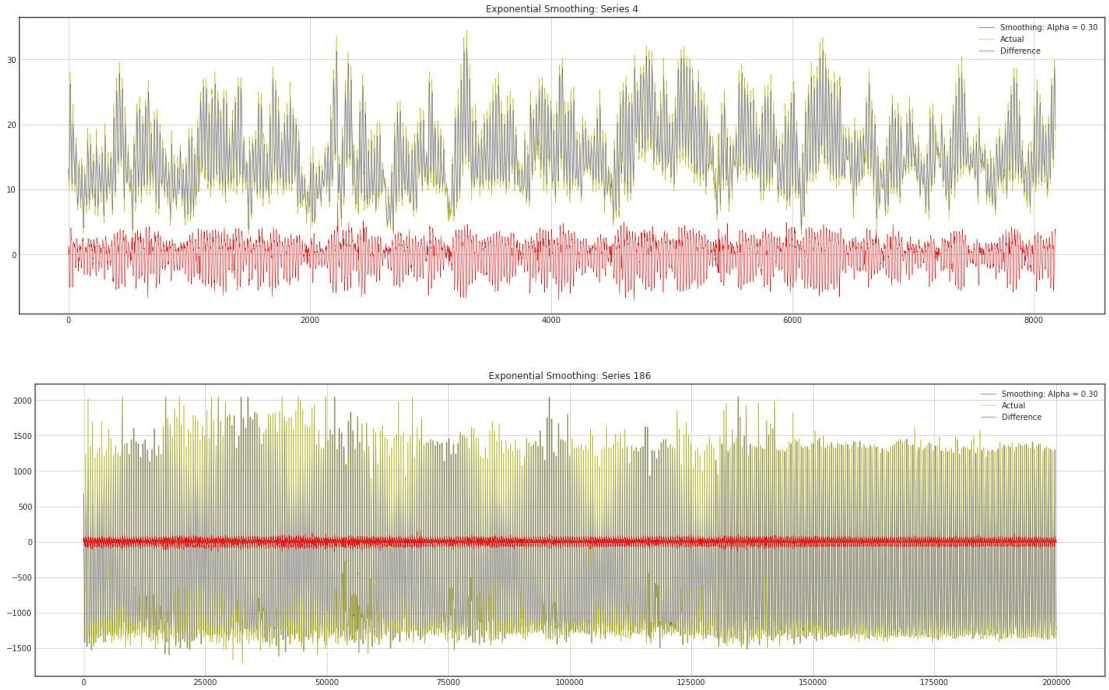


Figure 2: Examples of exponential smoothing process

As figure 2 shows, the new data in red, which is the difference between the original series and the smoothing series, will be stationary and be used in our models.

V. Model in Details

In this project, based on our exploratory data analysis and the work in data engineering, two broad approaches will be applied to complete the task: statistical approaches and deep learning based approaches. In the following sections, we will try several different models in these two broad approaches, compare their performance, and select the models that have the best performance.

A. Statistical Model: Matrix Profile

1. Basic Principle

The Matrix Profile is a relatively new data structure for time series analysis introduced in 2016 and developed by Eamonn Keogh of the University of California, Riverside, and Abdullah Mueen of the University of New Mexico. Some of the advantages of using the Matrix Profile are that it is domain-independent, fast, provides precise (approximate when needed) solutions, and requires only a single parameter.

The algorithm to calculate Matrix Profile uses the sliding window method. When the window size is M, the algorithm is:

1) Calculate the distance between window subsequence and the whole time series
2) Set an exclusion area to ignore unimportant matches
3) Update Distance profile with minimum value
4) Set the first nearest neighbour index

The distance calculation outlined above will occur n-m +1 times; Where, n is the length of the time series and m is the window size. Since the subsequence is extracted from the time series itself, an exclusion region is needed to prevent unimportant matches, such as regions that are very similar to the subsequence, generally asking before and after 1/2 of the current window index.

After calculating the Matrix Profile, the first K motifs and discards can be easily obtained. Motifs refer to repeated patterns in time series. The smallest distance indicates that the sequence is more similar to motifs, and the largest distance indicates more abnormal discords, where discords are just the anomalies we're targeting.

2. STUMPY

In our project, we use the STUMPY library to compute Matrix Profile. First, we need to divide the time series into multiple windows, and then use STUMP function to calculate the matrix profile of each window. For the window size, we set the associated parameters:

Parameters	Value
Minimum window size	50
Maximum window size	800
Growth rate	1.2

Table1: parameters in Matrix Profile

From above we can get all the Window size：

50	60	72	86	103	124	149	179	214	257	309	309	371	445	534	641	770

Next, we compute the matrix profile of each window using STUMP. The STUMP function in package STUMPY can efficiently compute a vector that stores the z-normalized Euclidean distance between any subsequence within a time series and its nearest neighbour. Practically, the use of the function STUMP optimized the traditional complex computation of matrix profile, it is only interested in storing the smallest non-

trivial distances from each distance profile, which significantly reduces the spatial complexity to O(n). (As figure 3 below shown)

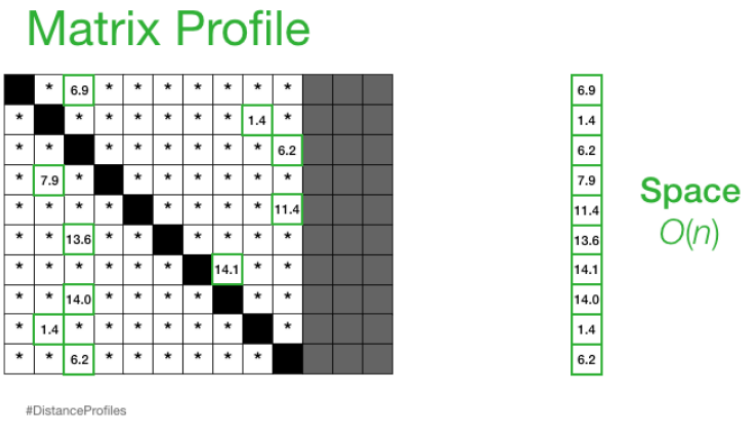


Figure 3: spatial complexity of Matrix Profile in NUMPY

We can now plot this matrix profile underneath our original time series (as the figure 4 below shown). And, as it turns out, a reference subsequence with a small matrix profile value (i.e., it has a nearest neighbour significantly "close by") may indicate a possible pattern while a reference subsequence with a large matrix profile value (i.e., its nearest neighbour is significantly "faraway") may suggest the presence of an anomaly.

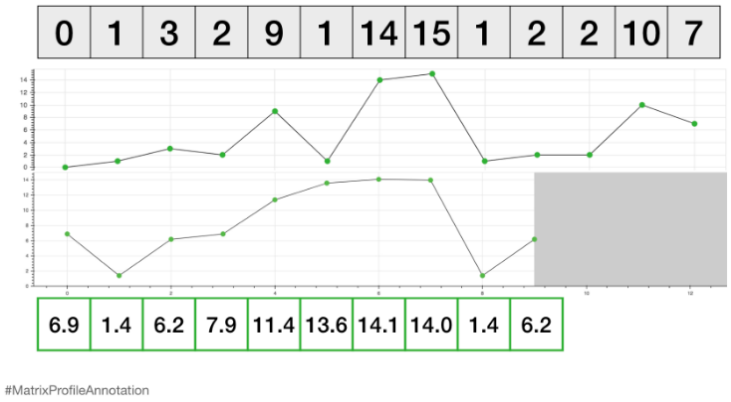


Figure 4: Matrix Profile value

3. Peak-to-Peak Algorithm

In the traditional Matrix Profile method, we can pick the window with maximum Matrix profile value, compare with the other maximum Matrix profile value of different window size, and get the index of the largest value as the anomaly. However, in our model, Peak-to-peak methods are combined to pick the anomaly. In the division of each window size, we set the maximum Matrix profile value as Peak1, mask the points in the positive and negative window size range near the peak value, as figure shown below, we find the remaining maximum Matrix profile value as Peak2, and calculate its difference as the rate. Besides comparing with other maximum Matrix profile values

of different window size, we also need to compare this rate of different window size. Get the index of both largest value and largest rate as the anomaly.

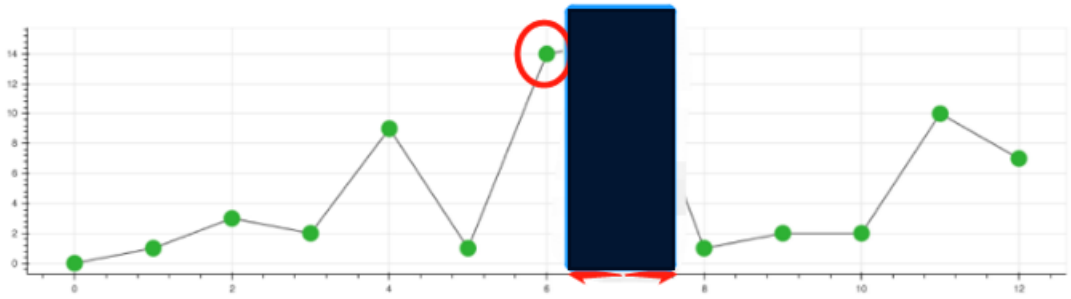


Figure 5: Peak-to-peak algorithm

B. Deep Learning Model: LSTM

1. Basic Principle

LSTM (Long Short Term Memory) networks were presented in the paper Hochreiter and Schmidhuber in 1997 and the first work about LSTM networks was in 1997 by Hochreiter and Schmidhuber to solve the problem of long-term dependencies. This network is a variant of the RNN architecture. Because it is difficult to train the model using the traditional RNN architecture, Recent advancements demonstrate state of the art results using LSTM (Long Short Term Memory). An LSTM cell has 5 essential components which allows it to model both long-term and short-term data: the cell state, hidden state, input gate, forget gate and output gate. One critical advantage of LSTMs is their ability to remember from long-term sequences. The LSTM architecture was able to take care of the vanishing gradient problem in the traditional RNN.

The LSTM network has the ability to remove and add information to the cell state, where the state of the cell can be compared to a conveyor belt, shown in figure 6 below, it passes through the enter chain, with only minor linear interactions. In LSTM, this process is regulated by special structures called gates.

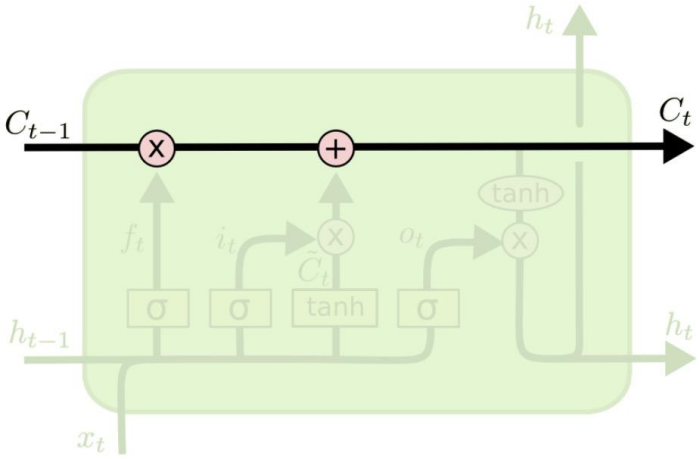


Figure 6: State of the cell

The gate is a mechanism that allows information to be passed selectively. It consists of a sigmoid layer (activation function-sigmoid) and a point multiplication operation. The output of the sigmoid layer is a number from 0 to 1, which determines the level of transmission. Zero means" don't miss anything "and one means “skip all”. The cell has three gates that control its state. The first gate decides which information to remove from the cell state. This decision is made by the sigmoid layer. The second gate decides which new information to write to the cell state. This stage is divided into two parts. First, the sigmoid layer, called the input gate, decides which values need to be updated. The tanh layer then creates a vector of New CT ' candidate values that can be added to the cell state. The third gate decides what we’re going to output. This output will be based on our cell state but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we’re going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

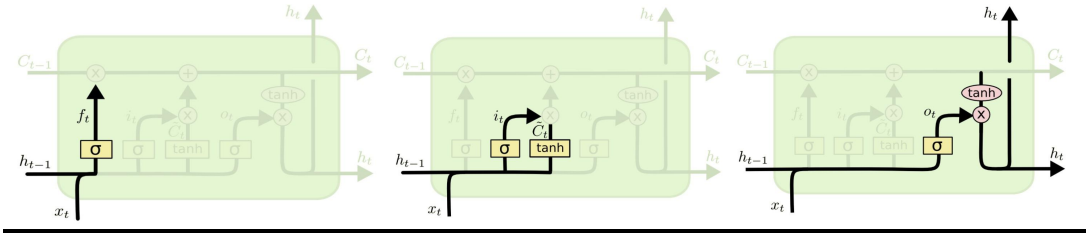


Figure 7: Gate in LSTM

2. Parameter Settings

Therefore, we built the LSTM model based on TensorFlow architecture, and our parameter settings are as follows:

Parameters	Parameters we set in LSTM
Neuron	50
Window size	1
Dropout	0.2
Dense	1
Activation	linear
Loss function	MSE
Optimizer	Adam

Table 2: Parameter settings in LSTM

It should be mentioned that we use the optimizer Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Because this method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning.

3. Early Stopping

Note that in some series, the training loss will increase in some series if training is completed in all 20 epochs, the early stopping method is necessarily introduced. This method can not only prevent the over-fitting of our model, but also shorten the time cost.

The early stop method is a widely used method and is better than the regularization method in many cases. Its basic meaning is to calculate the performance of the model on the verification set during training. When the performance of the model starts to decline on the verification set, the training is stopped, to avoid the problem of overfitting caused by continued training.

We implement the early stopping by *patience* parameter provided by Keras. Patience is set as three in our model, which means the training process will stop if the loss is not decreasing in 3 consecutive epochs. Figure 8 shows the loss dynamics in a complete 20-epoch training (left) and in a successful early stopping example at the 4th epoch (right).

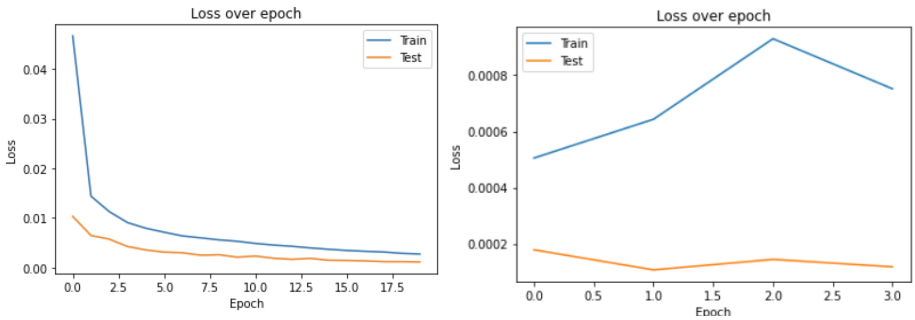


Figure 8: State of the cell

Finally, we can get the predicted value according to the model we built. By calculating the absolute values of difference of predicted value and actual value, we can find the index of the maximum absolute value, which is the anomaly.

C. Hybrid Model

Finally, we will adopt the method of combining Matrix Profile with LSTM. Because Matrix Profile runs at a faster speed and results are relatively stable, we will base on the Matrix Profile algorithm. For the deficiencies of the Matrix Profile algorithm, we will supplement the LSTM algorithm. The rule will be discussed in the next section.

VI. Performance

A. Performance of Matrix Profile

Because we have different types of time series, it is found that the Matrix Profile method is very stable in the processing of most time series but has defects in the processing of some types of time series. The specific defects are as follows: index is equal to 0.

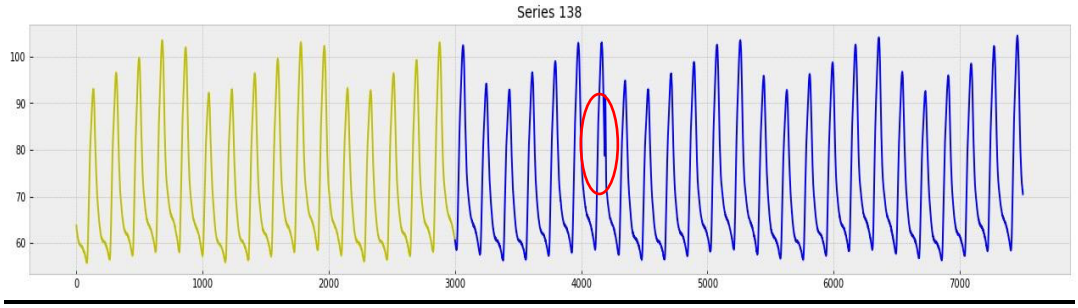


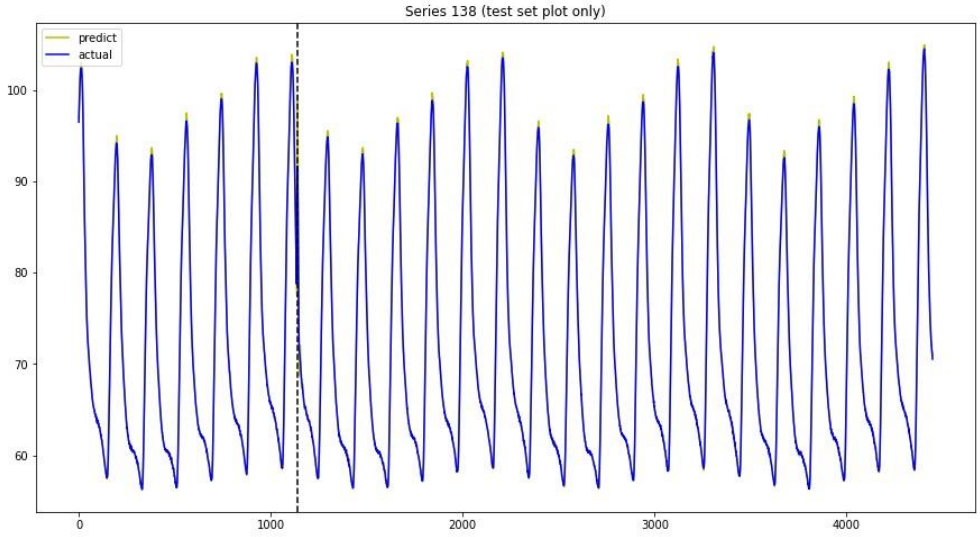
Figure 9: A bad example result of Matrix Profile

As can be seen from figure 9, between 4000 and 5000, there exists a partial large fluctuation range, which is circled in red and should be anomalies. However, when the Matrix Profile algorithm deals with a partial large fluctuation range, the index of anomalies will be equal to 0, and anomalies cannot be obtained correctly. Therefore, the Matrix Profile method is not applicable in this case.

In addition to the occurrence of zero value, the anomaly index points obtained by the Matrix Profile algorithm will be even before the split between the training set and the test set, which means the anomaly index is on the train set. However, all indexes obtained by us should be on the time series of the test set. So, the Matrix Profile method is not applicable in this case as well.

B. Performance of LSTM

The resulting index in LSTM does not have a zero value and a value before split. However, The LSTM has a fatal flaw: it is too slow. Although the early stop method is added to make the algorithm run faster, it still runs much slower than the Matrix Profile method. The performance examples of the LSTM are in figure below.



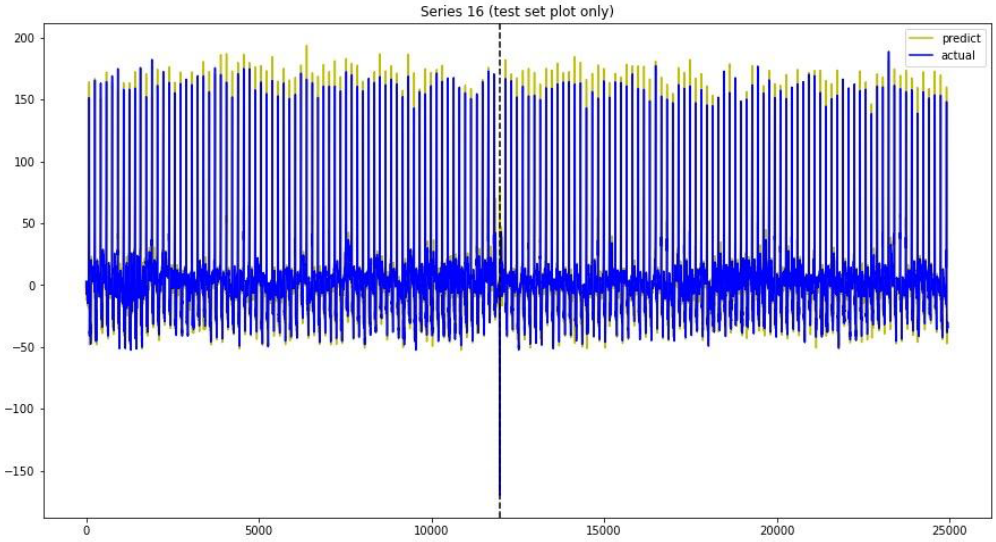


Figure 10: Example results in LSTM

C. Hybrid Model Performance

To conclude, the hybrid rule will be: Matrix Profile result will be recorded if it is not zero or before the split, or LSTM result will be submitted. Our comprehensive model is obviously superior to Matrix Profile and LSTM in performance. There is no zero value on the result, and there is no pre-split data. It is also superior to LSTM in running speed. Please refer to the file "proj-group24-result.csv" for our detailed results.

VII. KPI Dataset

A. Data Preprocessing

The data is 1-dimensional time series and can be grouped by 'KPI ID'. We firstly split the data into 29 datasets. And here are some samples shows.

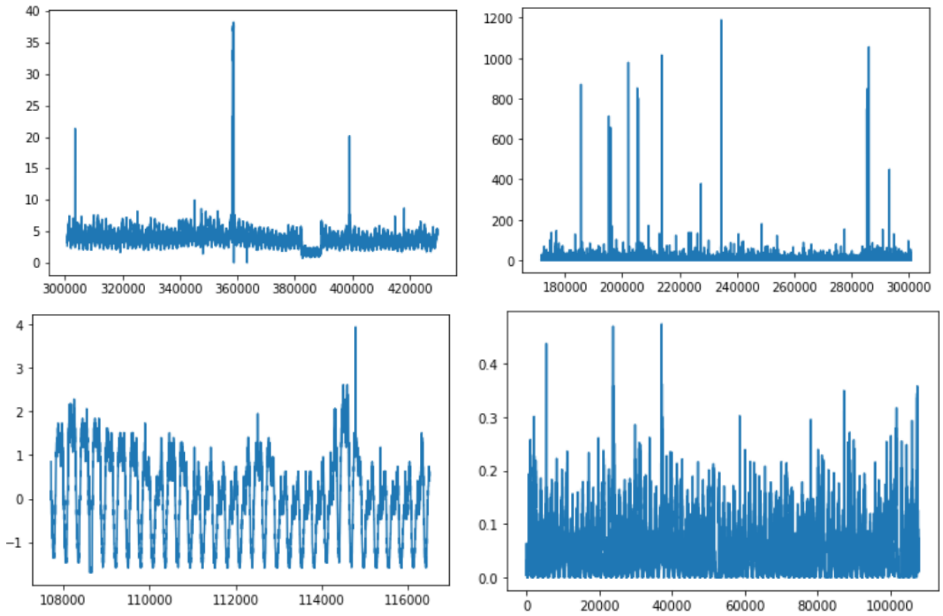


Figure 11: data samples of KPI

B. Model Construction

TCN (Temporal Convolutional Networks) is a time series convolutional network, which was proposed in 2018. In the past, once the sequence or time series data was mentioned, people usually use RNN models and its variants LSTM, GRU, etc. Many works have shown that it is difficult to find new models in the RNN framework, and its effect can surpass LSTM in many tasks. However, outside the RNN framework, the author of the paper we learned showed the use of CNN derived TCN can easily achieve effects that exceed LSTM and GRU in many tasks. And the model contains the following sections:

- Causal Convolutions

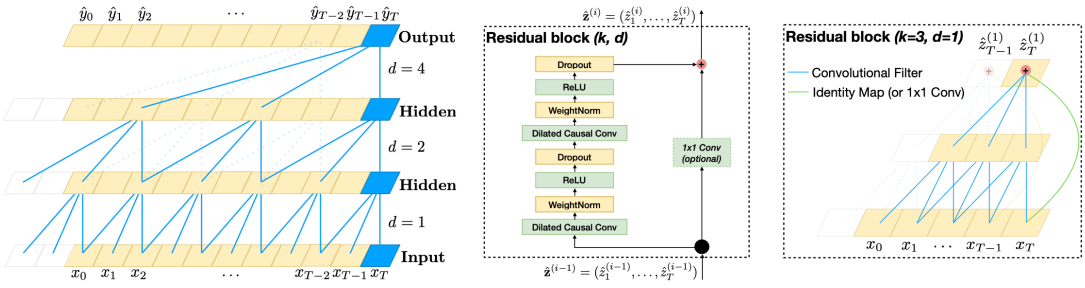
The causal convolutions use in this network are dilated convolutions that enable an exponentially large receptive field, and using the following function we can get the dilated convolution operation.

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}$$

- Residual Connections

This effectively allows layers to learn modifications to the identity mapping rather than the entire transformation, which has repeatedly been shown to benefit very deep net- works.

And the whole network architectural elements are shown below as well as our code.



(a) (b) (c)

Figure 12: architectural elements in TCN

Reference:

[1] https://github.com/Yorko/mlcourse.ai/tree/master/jupyter_english/topic09_time_series

[2] https://github.com/Yorko/mlcourse.ai/tree/master/jupyter_english/tutorials

[3] https://github.com/intellygenta/KDDCup2021

[4] https://stumpy.readthedocs.io/en/latest/Tutorial_The_Matrix_Profile.html
[5] Bai S, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[J]. arXiv preprint arXiv:1803.01271, 2018.