International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-025-02782-y

ORIGINAL ARTICLE

Patchlinear for single-step forecasting and anomaly detection

Min-Hua Zheng[1,2] · **Shi-Ying Lan**[1] · **Wan-Lei Zhao**[1] · **Jie Zhao**[2]

Received: 12 November 2024 / Accepted: 12 August 2025

Abstract
The trend forecasting and anomaly detection are the two fundamental tasks on a time series. They become increasingly important due to the popularity of AIOps in various scenarios. In the last few decades, increasingly complicated models have been proposed one after another for the trend forecasting and anomaly detection on the time series. Whereas, they are either ineffective or infeasible as they are too complicated to be deployed in real scenarios, where the online training is barely feasible. In this paper, an effective online method both for single-step forecasting and anomaly detection is proposed. In contrast to the existing methods that employ complicated models, the core component in our method is a two-layer linear network. Although simple, it already outperforms many state-of-the-art methods, such as DLinear and iTransformer on the single-step forecasting task. By the integration of distance significance which detects anomalies by referring to the recent history of the time series, our method also shows considerably superior performance on the anomaly detection task over most of the state-of-the-art methods such as FCVAE and AnomalyTransformer.

Keyword Time series · Single-step forecasting · Anomaly detection · Linear neural-network

1 Introduction

In the era of big data, a huge amount of time series data are generated each day from various sources, such as finance, IT security, medical, web services, social media, and geological information systems. The processing and analysis over such varieties of time series are mostly domain-specific. Let $X = \{x_1, \cdots, x_t, \cdots, x_n\}$ be a time series. There are two basic processing requirements that are shared in common by various applications. Firstly, it is expected that we can forecast the future trend of X. Namely, we want to know the future timestamps $x_{t+1 \cdots t+\tau}$ given $x_{1 \cdots t}$ are known. Secondly, we would expect to know whether the current status x_t is normal. This is particularly important for the key performance indicators (KPIs) time series. They basically reflect the health of a system. The abrupt changes occur when anything abnormal happens. These status undergone abrupt changes are called anomalies. The detection of these anomalies is critical to maintaining the health of a system. The timely alarm on these anomalies is expected to trigger human intervention or manual diagnosis of the system. This task is widely known as anomaly detection.

Time series forecasting and anomaly detection are the fundamental tasks in the time series analysis, whereas neither of them is trivial to address. There are two major challenges. First of all, the time series are mixed with anomalous status in practice. It is laborious and error-prone to annotate these anomalies manually. It is, therefore, unrealistic to adopt a fully supervised method to perform the detection. The existence of anomalies impacts the performance of forecasting as well. Moreover, due to the diversity of real-world time series, it is impossible to build a general model that performs well on all time series. Instead, the model should be adaptive to different time series. Because of the concept drift [47] in one time series, the model should run online as well, which allows it to be adaptive to the trend transition across different spans of one time series.

On the one hand, these two problems are old topics in the sense that the early research on the forecasting issues could be traced back to nearly one century ago [12]. In such a long period, classic methods such as ARIMA [4], Kalman Filter [14], and Holt-Winters [19] are proposed one after

✉ Wan-Lei Zhao
wlzhao@xmu.edu.cn

[1] Department of Computer Science and Engineering, Xiamen University, Xiangan Campus, Xiamen 361104, Fujian, China

[2] Boden AI Technology Ltd., High Tech. Zone, Ningbo 315048, Zhejiang, China

Published online: 31 August 2025

another. The implementations of these classic algorithms are found in the recent packages Prophet [43] and Hawkular [15]. On the other hand, new light has been shed on this century-old subject due to the need of artificial intelligence for IT operations (AIOps) [2, 10] in recent years.

Intuitively, anomaly detection can be addressed by statistical methods in either the time domain or frequency domain [3, 36]. Namely, a status is viewed as abnormal if it deviates from its mean so much that breaks the $3\text{-}\sigma$ rule. Unfortunately, such a solution is only effective for simple scenarios. Moreover, due to the lack of annotations on the anomalies, the fully supervised model either on forecasting or anomaly detection is not preferred. As a consequence, the mainstream methods perform the time series forecasting or reconstruction by unsupervised training. Given the forecasting or the reconstruction is sufficiently precise, the anomaly detection can be as easy as checking the degree of deviation of current status from the forecasted/reconstructed status [41, 44]. Before the introduction of deep neural networks, the representative forecasting methods are ARIMA [4], Kalman Filter [14], and Holt-Winters [19]. However, the forecasting precision of these methods is usually very low. They are unreliable to be based on to fulfill the detection. Although Prophet [43], recently developed by Facebook, performs significantly better, it still faces similar issues (as shown in Fig. 1b).

Due to the great success of deep neural networks in various domains, more and more methods built upon complicated deep models are proposed for both time series forecasting and anomaly detection. The sequential models such as recurrent neural network (RNN) [9] and long short-term memory (LSTM) [16] are adopted for forecasting as they are able to capture the sequential patterns across the different timestamps. Although encouraging performance is achieved [13], the forecasting results are not sufficiently good for anomaly detection (shown in Fig. 1c). In the recent studies [32, 44, 59], the Transformer has been adopted in time series forecasting and anomaly detection owing to its powerful correlation capability. As shown from Fig. 1d, the forecasting results from the Transformer-based methods fit well to the normal status in the source sequence. However, due to the high complexity of the model, it is hardly deployable in real scenarios where the training resources are limited, while the time series to be monitored could be in the hundreds.

In this paper, we propose a simple neural network to perform single-step forecasting and anomaly detection jointly. Contrary to the decade-old trend in the literature, that proposes increasingly complicated models with deep neural network or Transformer [5, 32] for forecasting and anomaly detection, time series forecasting and anomaly detection are addressed by simple neural networks, which only consist of two-layer linear neural network. Concretely, this network predicts the next status with given history time series data, and then the anomaly detection is carried out based on the prediction. The merits of our method are at least threefold.

- A simple but effective network is proposed for single-step forecasting. Its performance is on the same par as state-of-the-art Transformer-based methods while it is considerably efficient.
- Different from many existing works in the literature, the anomaly detection module is integrated with the forecasting module, which provides a universal solution for two fundamental tasks. Moreover, it shows the best performance across most of the evaluation benchmarks for the two tasks.

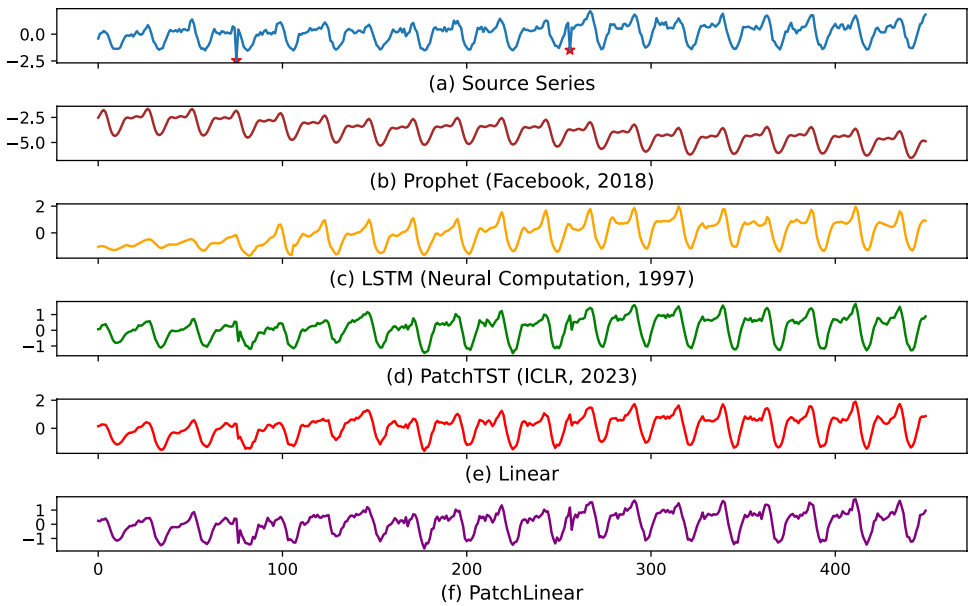


Fig. 1 A time series and the forecasting results of it from Prophet, LSTM, PatchTST, linear neural network, and the proposed PatchLinear, respectively. The anomalies in the source series are marked with red dots

- Furthermore, this method can be deployed online and runs in real-time.

The remainder of this paper is organized as follows. In Sect. 2, we review the related works about anomaly detection and time series forecasting. In Sect. 3.2, the two-layer linear network designed for the single-step forecasting task is introduced. In Sect. 3.3, the proposed anomaly detection approach is presented. In Sect. 4, the effectiveness of our approach is evaluated on four univariate time series datasets. Finally, we conclude our paper in Sect. 5.

2 Related works

2.1 Time series forecasting

The aim of time series forecasting (TSF) is to predict the status of future τ timestamps when the history timestamps $x_{1\ldots t}$ are provided. The predictor is expected to capture the seasonal patterns and trend, and to be robust to concept drift in the meantime. The forecasting can be long-term when $\tau \gg 1$ or single-step when $\tau = 1$. For the long-term forecasting task, the method is required to focus on the long-term trend of the time series. While for the single-step forecasting task, the method is required to capture the short-term patterns in the time series. For this reason, the methods designed for long-term forecasting are not necessarily effective on the single-step forecasting problem, and vice versa.

In the literature, we can see methods proposed either for long-term forecasting or for single-step forecasting. Before the emergence of deep models, single-step forecasting was addressed mainly by methods from statistics or signal processing, such as ARIMA [4], Kalman Filter [14], and Holt-Winters [19]. The recent method Prophet [43] can be viewed as the enhancement over these traditional methods, in which the time series is decomposed into different components, and the different components are fit by different models. In general, these methods are computationally efficient, and they are well-suited for performing forecasting on the stationary time series. However, the time series produced in various contexts are usually non-stationary. The correlation between different timestamps is non-linear. Very poor performance is observed on such time series by the traditional methods. Before the emergence of deep neural network, evolutionary computing methods [39] like PSO [20], GA [17] and its variants [6, 40] have been employed to address the time series forecasting task and show considerable improvement over the traditional methods. However, they also face some notable drawbacks, such as high sensitivity to the hyperparameters and the initial conditions.

Owing to the high model complexity, the deep neural networks such as the LSTM [16] are expected to have the better capability in capturing the temporal patterns. In the last decade, there have been various forecasting methods built upon LSTM [13]. In order to discover the most relevant feature to the prediction, the attention strategy has been integrated into the RNN network [35]. However, these types of methods are sensitive to anomalies and noises. In order to boost the performance, ensemble learning is introduced into forecasting [51, 54]. However, such methods induce significantly higher computational costs.

Transformer [45] based models have shown great success in natural language processing and computer vision. Recently, several long-term forecasting methods based on Transformer have been proposed. Namely, they are Informer [59], Autoformer [48], Pyraformer [28], Triformer [8], FEDformer [60], and PatchTST [32]. Informer in [59] alleviates the induced computational complexity by Transformer model when applied in long-term forecasting. The variants of transformer [28, 32, 48, 60] integrate extra features into the model to either improve its performance or efficiency. The common issue latent in these transformer [45] based models is their vulnerability to the concept drift since they cannot be deployed online. Recently, the long-term forecasting methods based on Transformer have been challenged by a linear model [55]. As pointed out in the paper, a simple linear network outperforms Transformer-based methods such as Informer, Pyraformer, Autoformer, and FEDformer on the long-term forecasting task. The experiments in [32] confirm this discovery. In our paper, time series forecasting is designed to forecast the next incoming status, and is also used as the supporting block for the anomaly detection.

2.2 Anomaly detection in time series

Because the single-step time series forecasting only predicts the expected status for a timestamp *t*, the anomaly detection can be achieved by checking the deviation of x_t from the predicted status $\bar{x}_t$ at timestamp *t*. If $\bar{x}_t$ deviates from x_t too much, it is viewed as an anomaly. As a result, the aforementioned forecasting methods are potentially anomaly detection methods as well. In the following, brief reviews of other types of methods are presented.

There are varieties of reconstruction-based methods in the literature, nevertheless, they are out of the same motivation. Since the status of anomalies are far from normal and have a rare occurrence, they will be reconstructed to normal status. Similar to forecasting-based methods, x_t is viewed as an anomaly when it is far from the reconstructed status $\bar{x}_t$. Traditionally, the reconstruction is fulfilled by PCA or SVD. In recent works, generative adversarial network

(GAN) [26, 57, 58], autoencoder (AE) [24], variational autoencoder (VAE) [5, 34, 42], and transformer [50] are widely used to represent the deep features, then check the deviation to detect anomalies. The deep learning methods outperform traditional methods considerably. However, due to the high model complexity, transformer-based methods are susceptible to overfitting. While GAN-based methods are unstable due to the potential mode collapse during the training. For these reasons, both transformer-based methods and GAN-based methods require anomaly-free time series for training, which are hard to collect in practice. Moreover, there is no consideration about the online deployment of the model in any of these methods. Namely, these methods cannot be self-adaptive to the concept drift in the time series.

Apart from the forecasting-based and reconstruction-based methods, there are also detection methods based other theories. Dynamic spatial pyramid occupancy time-series model (DSPOT) is built on extreme value theory [38]. By integrating spatial information with temporal dynamics, DSPOT aims to detect anomalies and unusual patterns that may occur in both space and time. This method is only effective when the anomalies exhibit spatial and temporal dependencies. Spectral residual (SR) [36] performs anomaly detection in the frequency domain. The source time series is transformed into the frequency domain by the Fast Fourier transform. The significant residuals in the frequency domain correspond to the anomalies in the time domain. SR is very efficient and is effective in detecting the anomalies that deviates from normal significantly. It fails on the subtle anomalies which cannot be revealed in the spectral residue. Recently, the matrix profile (MP) [53] also has been adopted for anomaly detection. The x_t is viewed as an anomaly when the distance of the subsequence containing x_t to its closest subsequence is abnormally larger than the average distance. Although simple, MP shows promising detection performance. In contrast to the methods based deep models, both SR and MP can be deployed online. Similar to SR, MP fails on the subtle anomalies.

In this paper, we address the time series single-step forecasting and anomaly detection jointly. In contrast to building complicated models in many existing works [5, 32], a simple two-layer linear neural networks is adopted for the single-step forecasting. Based on the forecasted value $\bar{x}_t$ at timestamp t, the judgment about whether x_t is an anomaly is made.

3 Proposed method

3.1 Overview of the framework

Given a time series $X = \{x_1, \cdots, x_t, \cdots, x_n\}, x_t \in R^d$, where d is the time status dimension. Specifically, X is a univariate time series when $d = 1$, while X is a multivariate time series when $d > 1$. In the forecasting task, we are going to forecast x_{t+1} given $x_{1\cdots t}$ are provided. In the the anomaly detection, we are going to judge whether x_t is abnormal given $x_{1\cdots t-1}$ are provided. Our anomaly detection is based on the output from forecasting, namely the forecasted status $\bar{x}_{t+1}$ will be used for anomaly detection at timestamp $t + 1$. For the convenience of our discussion, the anomaly detection is performed on timestamp $t + 1$ given $x_{1\cdots t}$ are known from now on. In practice, this is easy to achieve to hold the forecasted status for the anomaly detection in the next timestamp.

Inspired by the discovery from [55], we attempt to fulfill the single-step time series forecasting by a linear neural networks. Thereafter, the anomaly detection is performed based on the forecasted status $\bar{x}_{t+1}$. The whole framework for single-step forecasting and anomaly detection is shown in Fig. 2. There are basically two processing pipelines in the framework. On the bottom pipeline in the figure, the time series are fed into a two-layer linear networks after a few steps of preprocessing. The network will first make the forecasting for timestamp $t + 1$. On the one hand, the forecasted status $\bar{x}_{t+1}$ is one of the outputs of our framework. On the

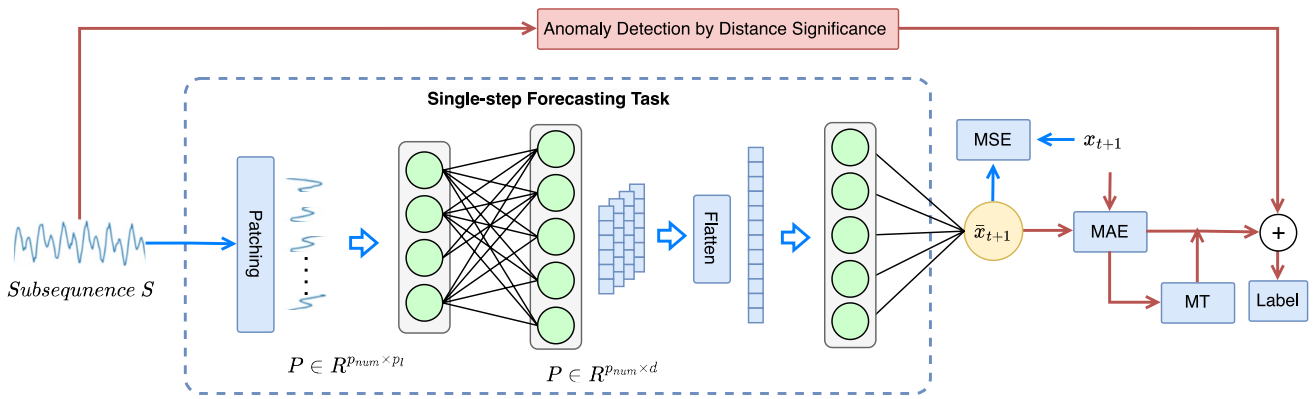


Fig. 2 The framework of the PatchLinear model. The flow in blue is responsible for the time series forecasting. The result from forecasting is fed into the flow in red for anomaly detection

other hand, it is used as a weighted vote for the anomaly detection. On the top pipeline, the time series are fed into an unsupervised anomaly detection method. The anomaly judgment is made based on the "distance significance" of x_{t+1} from the historical status $x_{1\cdots t}$. The final judgment on whether x_{t+1} is an anomaly is made by a weighted vote from two pipelines.

3.2 Single-step time series forecasting

In the forecasting task, given a time series X, we are going to forecast x_{t+1} when $x_{1\cdots t}$ are provided. It is possible to adopt transformer [32, 48, 59] to fulfill the forecasting. However, linear neural network such as DLinear [55] is preferred for at least two reasons. First of all, self-attention mechanism in transformer makes it permutation-invariant. It is good in language translation task, however it is harmful for time series forecasting, in which the strict sequential order matters. Moreover, the quadratic time and memory complexity makes self-attention mechanism an unaffordable operation in this context which typically requires real-time performance. In contrast, the light-weight linear neural network already demonstrates excellent performance in long-term time series forecasting task than many Transformer-based methods [55]. Therefore, in our single-step time series forecasting task, the linear neural network is adopted.

The length of time series could be arbitrary, while the size of neural network input is fixed. Following the common practice in the literature, the time series is cut into subsequences. Given the arrived t timestamps of time series X, window size w_1, and stride 1, time series is transformed into a set of subsequences $\varphi = \{S_1, \cdots, S_k, \cdots, S_{t-w_1+1}\}, S_k \in R^{w_1}$. Please note that X is z-score normalized before it is cut into subsequences φ. Then following the practice in [32], the subsequence is further divided into patches to capture the patterns within the small temporal window. Namely, S_k is cut into overlapping patches with window size p_l and stride s. As a result, a subsequence S_k is patched as $P_k = \{p_1, \cdots, p_j, \cdots, p_{num}\}$, where $num = \frac{w_1 - p_l}{s} + 1$. As revealed later, such kind of patching allows the network to extract detailed feature representation from each subsequence. These features capture local patterns or temporal characteristics within shorter time frames. The patched subsequence is first normalized by a reversible instance normalization, as proposed in previous work [21], and then is fed into a two-layer fully connected linear network for the single-step forecasting task. The network is formulized in Eq. 1. Given $\bar{x}_{t+1}$ is the forecasted status at timestamp $t+1$, the loss of the forecasting network can be simply defined by mean squared error (MSE), which is given by Eq. 2. The whole forecasting network is shown inside the dashed bounding-box in Fig. 2.

$$\bar{x}_{t+1} = W_2 \times (Flatten(W_1 \times p_j + b_1)) + b_2, \ j \in [1, num] \tag{1}$$

$$\text{Loss} = \sum_{t=1}^{n} (\bar{x}_{t+1} - x_{t+1})^2 \tag{2}$$

As revealed later in the experiments, the forecasting performance of this simple network is on the same par as the state-of-the-art method PatchTST in most of the cases. Moreover, due to the simplicity of our method, it is cheap to be trained and deployed as an online model, which is able to adapt to the concept drift easily.

Discussion In the above network, a sliding window is adopted when we cut subsequence S_k into overlapping patches. Thereafter, the patches are fed into a two-layer linear network. Since the patches are produced from a subsequence with a sliding window, the first layer is comparable to applying a 1D-convolution over S_k. It therefore captures the correlation between timestamps within a short temporal range. The second linear layer performs linear mapping between the output of the first layer and the $\bar{x}_{t+1}$. Compared to the first layer, it covers the whole subsequence S_k. It, therefore, has a wider receptive field. As a result, it is able to mine the correlation between timestamps in a much wider range. As pointed out in [27], most of the local timestamps are linear dependent, it is sufficient to fulfill the single-step prediction by a layered linear-mapping [55].

Given the single-step time series forecasting is sufficiently precise, it is easy to judge whether the status on a timestamp x_{t+1} is anomalous by referring to the forecasted value for this timestamp. In the next section, we are going to show how the forecasting network presented in this section serves as a major component for the anomaly detection task in our overall framework.

3.3 Anomaly detection

In the anomaly detection task, we are going to judge whether x_t is abnormal when $x_{1\cdots t}$ are known. Our anomaly detection is based on the output from forecasting, namely the predicted status $\bar{x}_{t+1}$ will be used for anomaly detection at timestamp $t+1$. So there is one timestamp delay between the forecasting task and the detection task. For the convenience of our discussion, the anomaly detection is performed on timestamp $t+1$ given $x_{1\cdots t}$ are known from now on. In practice, it is straightforward to save forecasted status until next timestamp arrives for anomaly detection task.

Thereafter, the anomaly detection is performed based on the predicted status $\bar{x}_{t+1}$. The whole framework consists of single-step forecasting and anomaly detection, as shown in Fig. 2. There are two processing pipelines in the framework. On the bottom pipeline in the figure, $\bar{x}_{t+1}$ is obtained by the single-step forecasting module. On the one

hand, the forecasted status $\bar{x}_{t+1}$ serves as one of the outputs of our framework, namely single-step forecasting. On the other hand, the deviation of x_{t+1} from the forecasted status $\bar{x}_{t+1}$ is used for the downstream anomaly detection task. Herein, we adopt the principle that is shared by prediction-based anomaly detection methods "*the anomalies are unpredictable*" [1].

In order to boost the detection performance, another method based on the proposed distance significance is integrated into the framework, which is shown as the pipeline on the top part in Fig. 2. In the rest of this section, we are going to present two integrated detection methods in detail.

3.3.1 Moving threshold

In general, "*the anomalies are unpredictable*", the forecasting component introduced in the previous section will only forecast the normal status of timestamp $t+1$. It is straightforward to judge whether status x_{t+1} is anomalous by referring to the forecasted status $\bar{x}_{t+1}$. We select the mean absolute error (MAE) to measure the deviation as anomaly score. Then, the key issue is to decide to what degree that x_{t+1} deviates from $\bar{x}_{t+1}$ that it should be viewed as an anomaly. Intuitively, one could set a fixed threshold to dictate whether x_{t+1} is an anomaly. Several anomaly detection methods compute an anomaly threshold after they get all anomaly scores for a dataset [44, 46]. However, this is unrealistic in practice. First of all, only the status before timestamp $t+1$ are known. One cannot estimate the threshold based on future status. On the other hand, one cannot simply fix the threshold due to the concept drift. To address this issue, we propose to update this threshold dynamically.

Given μ and σ are the mean and standard deviation of a subsequence respectively, the threshold for the current status is defined as

$$T_{t+1} = \mu + k \cdot \sigma, \tag{3}$$

where k is a hyper-parameter. In Eq. 3, μ and σ are estimated from a subsequence. At different stages, μ and σ are estimated in different ways. In principle, they are estimated from the subsequence that is close to timestamp $t+1$. At the initial stage, due to the insufficient number of time status data for prediction and detection, we have to rely on the μ and σ computed during the training stage. Meanwhile, in order to be adaptive to continuously arriving new status, T_{t+1} is set to be the maximum among the one estimated from the training sequence and the one estimated from the first *30* timestamps from the current subsequence. When sufficient number of new status arrive during the test stage, the deviations of the status from its forecasted status at the recent c_w timestamps are cached. The μ and σ are, therefore, estimated from these cached statuses. At this stage, T_{t+1} is set to be the maximum among the estimated from the cache and the first *30* timestamps estimated from the current subsequence.

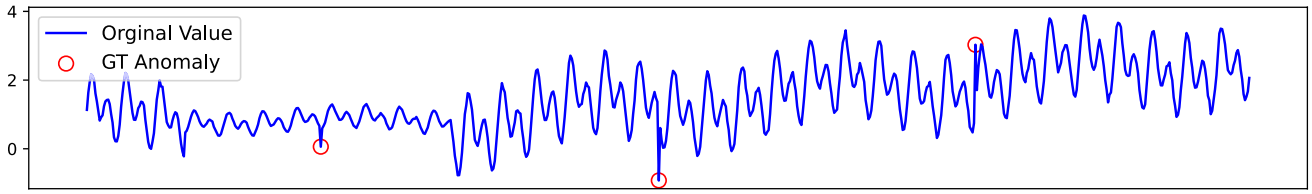
Figure 3d shows the moving threshold derived from arrived time subsequence. Due to the good forecasting performance (as shown in Fig. 3c), most of the anomalies can be easily identified by the moving threshold.

The above anomaly detection based on MAE is effective in detecting anomalies on a stationary time series. Nevertheless, it fails on the non-stationary time series. A typical scenario is given in Fig. 3d. This strategy fails to detect the first anomaly on the time series, where the anomaly is insignificant compared to the variations across the arrived status (status on the left of the first anomaly). To address this issue, another method called distance significance is proposed, which is detailed in the next section.

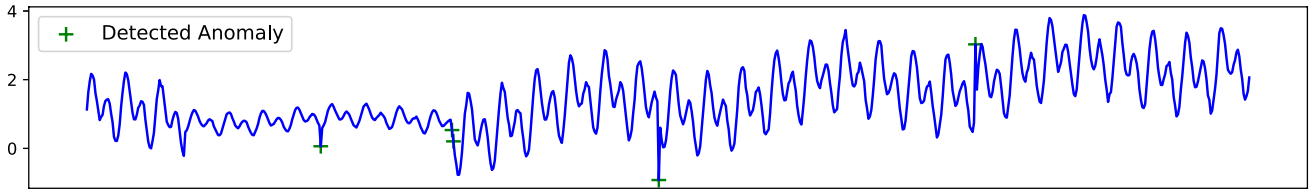
3.3.2 Distance significance

In the above method, anomaly detection on a timestamp is achieved by checking the deviation of real status from the forecasted status. Nevertheless, anomaly detection based solely on the single-step forecasting cannot necessarily lead to good performance. On the one hand, a status is viewed as anomaly usually by referring to its neighboring status. All these consecutive status could be in small value when they are in the valley of a curve. On the other hand, the moving threshold is largely the average of recent status, which could be large due to the peaks in the curve or the potential anomalies. For this reason, the anomalies in small value could be overlooked. Such issues are illustrated in Fig. 3d. To address these issues, we integrate the distance significance (DS) into the detection model, which contrasts the difference of a status against the best matched subsequence in the history. This strategy has been widely adopted in various fields. For instance, the meteorologist judges whether the temperature of a certain day is abnormal by referring to the historical temperature data of that day in history. In particular, he will compare with the year when the temperature data are the best-match in a small period before that day. Based on this idea, our second anomaly detection strategy is conceived. There are basically two steps in this strategy. Namely, we first search for the best-match subsequence from the observed time series. Then, we measure the significance of the difference between the current status and the best-matched subsequence. If they are significantly different, current status is deemed as an anomaly.

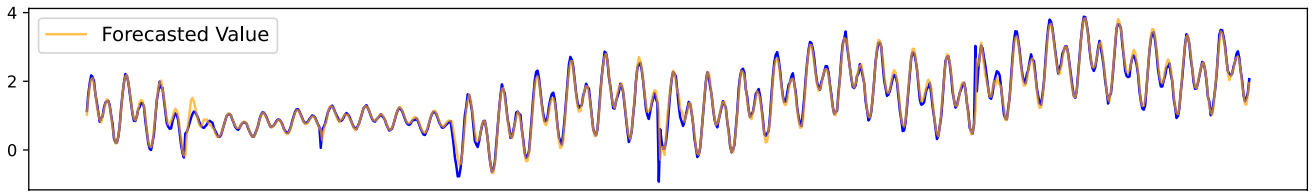
Given a time series X, the subsequence to be judged $X_{t-m+1,m}$ is defined as a continuous interval of length m starting from the position $t-m+1$, i.e., $X_{t-m+1,m} = \{x_{t-m+1}, x_{t-m+2}, \ldots, x_{t+1}\}$. Similar to the



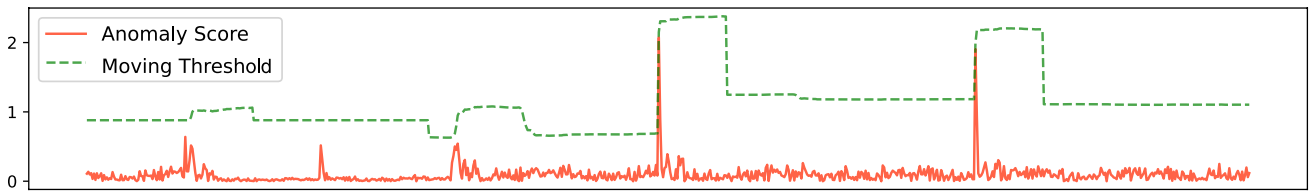
(a) Time series and its anomalies



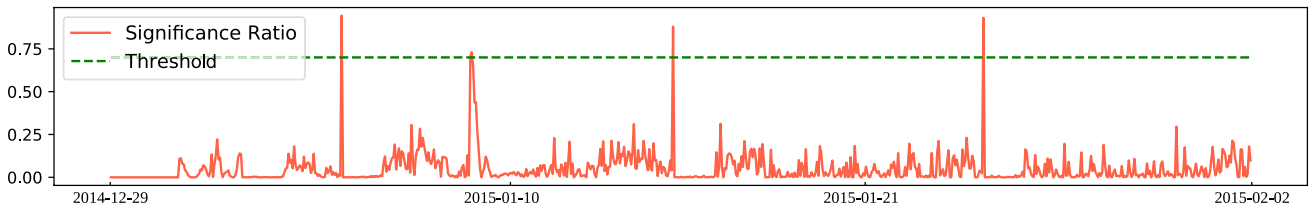
(b) Anomalies detected by our model, marked on groundtruth series



(c) The forecasted statuses by our method along with the original time status



(d) MAE score and the computed moving threshold



(e) Distance significance ratio

Fig. 3 The illustration of anomaly detection by different modules in PatchLinear. **d** Shows the anomaly detection based on the forecasting results in figure (**c**). **e** Shows the anomaly detection based on distance significance

forecasting task, the time series X is cut into subsequences with a sliding window. The step size of the sliding window is 1 and the window size is m. Given a query subsequence $X_{i,m}$, its *Euclidean* distance to another subsequence $X_{j,m}$ is computed by

$$d(X_{i,m}, X_{j,m}) = \sqrt{\|(X_{i,m} - \mu_i 1) - (X_{j,m} - \mu_j 1)\|^2}, \quad (4)$$

where μ_i and μ_j are the mean of $X_{i,m}$ and $X_{j,m}$ respectively. Following the way of matrix profile [61], we can first rewrite Eq. 4 into

$$d(X_{i,m}, X_{j,m}) = \sqrt{m(\sigma_i^2 + \sigma_j^2) - 2(\langle X_{i,m}, X_{j,m}\rangle - m\mu_i\mu_j)}. \quad (5)$$

This distance computation between $X_{i,m}$ and $X_{j,m}$ can be sped up by the following Eq. 6.

$$\langle X_{i,m}, X_{j,m}\rangle = \langle X_{i-1,m}, X_{j-1,m}\rangle - x_{i-1}x_{j-1} + x_{i+m-1}x_{j+m-1}, \quad (6)$$

where σ_i, σ_j in Eq. 5 are the standard deviations of two subsequences.

In order to avoid trivial matches [31], an exclusion zone of $\frac{m}{2}$ before and after the subsequence in the time series is ignored. Given subsequence $X_{j,m}$ is the best-matched for the current sequence $X_{i,m}$, the timestamp *j* will be recorded for subsequence $X_{i,m}$. In order to speed up the matching procedure, we only keep the last c_w inner products, which means we only match the most recent c_w subsequences. In this way, the time complexity of the distance profile is reduced to $O(c_w{\cdot}m)$.

Once we find the best-matching subsequence for the current subsequence, we should proceed to evaluating the difference between them such that to judge whether the last timestamp (x_{i+m-1}) in $X_{i,m}$ is abnormal. This timestamp is exactly x_{t+1} that is aforementioned.[1] Given subsequence $X_{i,m}$ and its best-matching subsequence $X_{j,m}$, we aim to judge whether the last timestamp in subsequence $X_{i,m}$, namely x_{i+m-1} is an anomaly. To achieve that, the distance significance on timestamp $i+m-1$ is defined as

$$\hat{r}_i = \frac{((x_{i+m-1} - \hat{\mu}_i) - (x_{j+m-1} - \hat{\mu}_j))^2}{\sum_{w=m-l}^{m-1}((x_{i+w} - \hat{\mu}_i) - (x_{j+w} - \hat{\mu}_j))^2}, \quad (7)$$

where $\hat{\mu}_i$ and $\hat{\mu}_j$ are the means of the last *l* status of $X_{i,m}$ and $X_{j,m}$ respectively. Equation 7 measures the significance that the distance between last status x_{i+m-1} and x_{j+m-1} in contrast to a strip of distance between $X_{i,m}$ and $X_{j,m}$. As shown in Fig. 3e, we can see that if x_{i+m-1} is abnormal, $\hat{r}_i$ is unexpectedly high, and it is invariant to amplitude changes since it is a distance ratio relative to its best-matched subsequence. In Eq. 7, *l* is a parameter and is set to *30*, which is the same as the MAE score window. In practice, there may be more than one anomaly within $X_{i,m}$. As a result, *l* is recommended to be smaller than *m* when *m* is a large status, otherwise status $\hat{r}_i$ remains small even if an anomaly occurs. Once the distance significance ratio $\hat{r}_i$ is ready, an anomaly is detected when its $\hat{r}_i$ is above a hyper-parameter threshold $\eta \in (0, 1)$.

Table 1 The number of parameters across different layers in our network under two settings "hourly" and "minutely"

Granularity	Input w_1	Patching	Layer-1	Layer-2	#Para.
Hourly	24	$p_l = 12, s = 4$	$12 \rightarrow 64$	$(4 \times 64) \rightarrow 1$	1344
Minutely	120	$p_l = 12, s = 4$	$12 \rightarrow 64$	$(28 \times 64) \rightarrow 1$	4416

[1] For the convenience of presentation, we refer x_{t+1} as x_{i+m-1} across this section.

Nevertheless, it is possible there are anomalies in the best-match subsequence. In this case, it is no longer meaningful to treat the best-match subsequence as the reliable reference for the detection. To address this issue, the median absolute deviation (MAD) [25] is adopted to detect anomalies on a given subsequence. Similar to the distance significance, a status of a timestamp x_{i+m-1} is viewed as an anomaly when the D_{i+m-1} in Eq. 8 is higher than a fixed threshold $\theta \in (0, 1)$.

$$D_{i+m-1} = \frac{0.674 * |(x_{i+m-1} - \text{Median}(X_{i,m}))|}{\text{Median}(|(X_{i,m} - \text{Median}(X_{i,m}))|)}, \quad (8)$$

where function $\text{Median}(\cdot)$ returns median of an input subsequence. The factor *0.674* is an empirical constant from [25]. MAD judges whether the last timestamp in the subsequence is an anomaly based on the properties of the subsequence itself when the best-match subsequence is not reliable. It checks whether the last timestamp in the current subsequence deviates significantly from the median of the subsequence. Relying on the median instead of the commonly used mean, it becomes robust to extremely high or low status.

The complete detection framework is shown in Fig. 2. The first detection pipeline relies on the output from the single-step forecasting, which has been elaborated in Sect. 3.3.1. The second pipeline (pipeline on the top) relies on distance significance and median absolute deviation. Given a status x_{t+1}, it is viewed as an anomaly when it is detected by either of them. The source codes of our paper are available at https://github.com/wlzhao22/PatchLinear.

3.4 Complexity analysis

There are two pipelines in our anomaly detection framework. Namely, one pipeline is a two-layer linear network. Another essentially performs subsequence matching between the current subsequence and the historical ones. The complexity of the first pipeline is mainly determined by the scale of network parameters. The number of parameters in different layers is shown in Table 1. For time series in different granularities, two configurations are used. As shown from the table, the number of parameters is less than 5K, which is quite light-weight in contrast to 100K involved in the Transformer-based method [32].

For the pipeline of anomaly detection based on distance significance, the most intensive operation is the distance calculation between subsequences. Given c_w and *m* are the cache length of historical series and the window size of current subsequence respectively, the time complexity is $O(c_w \cdot m)$. As a result, the time complexity of anomaly

detection is $O(t \cdot c_w \cdot m)$ when the length of a time series is t.

4 Experiments

4.1 Datasets and evaluation protocol

In this section, the effectiveness of our method in single-step forecasting and anomaly detection is evaluated on four benchmarks, KPI [2], Yahoo [52], NAB [23], and MSLU [18]. The brief information about these four datasets is summarized in Table 2. The KPI dataset is released by the AIOps Challenge Competition. The KPIs series are collected from various Internet companies, such as Sogou, eBay, and Alibaba. All the time series are minute-level, and the anomalies are annotated. The Yahoo dataset is released by Yahoo Lab. The dataset consists of real-world and synthetic time series. The real-world series are collected from the real traffic of Yahoo services. Both the real and synthetic series are hour-level. NAB is an open dataset released by Numenta. Following previous work [46], we apply 10 univariate time series from Twitter in this dataset. MSLU is derived from the MSL dataset, a multivariate benchmark for time series anomaly detection released by NASA, which recorded the status of the Mars rover. The first dimension of MSL is adopted in our experiment as a univariate time series for evaluation. For all of the datasets, each time series is evenly divided into two parts. The first part is used for training, and the second part is for test. For the subsequence used for training, the first 70% of timestamps are used for training, while the rear 30% of timestamps are used for validation.

In the single-step forecasting task, following the practice in [32], mean squared error (MSE) and mean absolute error (MAE) are adopted in the evaluation. The lower MSE and MAE, the better the forecasting performance. In the anomaly detection task, following the convention in the literature [36, 49], precision, recall, and F1-score are used in our evaluation. Usually, operators are more concerned about whether an anomaly can be successfully detected within an acceptable delay in real practice. Therefore, following the evaluation metrics in previous works and competition [2, 36, 49], the detection is viewed as a true-positive sample if an anomaly is detected in q timestamps after the anomaly is generated. Following the previous practice in [2, 5, 46], the hyper-parameter q is 7 in KPI, 3 in Yahoo, 150 in NAB, and 60 in MSLU, respectively. The higher the precision, recall, and F1 score, the better the detection performance. In addition, the average time cost of forecasting every timestamp by different methods are reported.

Table 2 The brief statistics on the four evaluation datasets

Dataset	#Series	#Timestamps	#Anomalies	Granularity
KPI	29	5,922,913	134,114 (2.26%)	Minute
Yahoo	367	572,966	3,896 (0.68%)	Hour
NAB	10	158,631	15,689 (9.89%)	Minute
MSLU	18	78,644	6,622 (8.42%)	Minute

4.2 Experiment setting

In this section, we set two versions of parameters for datasets with different time granularities. For minute-level datasets, matching window m, sequence length w_1, and threshold η are experimentally set to *1,440*, *120* and *0.4*, respectively. Specifically, due to the short length of the MSLU dataset, the matching window is set to 360. For hour-level datasets, m, w_1 and η are *48*, *24* and *0.7*, respectively. The cached window c_w, MAD threshold θ and score window l are set to 10 days, 0.5 and 30 for all datasets, respectively. All the experiments are performed on a PC with an Intel i7-8700 CPU @ 3.2GHz (12 cores) and 16GB of memory.

4.3 Performance on single-step forecasting

The performance of our method is compared with the representative time series forecasting and anomaly detection methods in the literature. For forecasting task, the conventional approaches ARIMA [4] and Prophet [43], classic methods based on neural network such as LSTM [16], GRU [7], and recent methods PAD [5], DLinear [55], iTransformer [29] and PatchTST [32] are also considered. ARIMA and Prophet are the classic methods for time series forecasting tasks. LSTM and GRU are the classic neural-networks used in time series forecasting task. The PAD is an integration of VAE and LSTM that is able to perform the forecasting task and anomaly detection task on the time series. DLinear, iTransformer, and PatchTST are recently developed and show superior performance on time series forecasting tasks.

The performance of the single-step forecasting task from our method and state-of-the-art methods is shown in Table 3. The performance from the classic methods such as ARIMA and Prophet is pretty poor across all the datasets since they are unable to capture the non-linear correlation between timestamps. Deep models such as GRU, LSTM, and PAD demonstrate satisfactory performance on KPI dataset, whereas perform poorly on the rest due to their vulnerability to the noises and anomalies. iTransformer performs well on the Yahoo dataset but gets unsatisfactory results on the KPI dataset, indicating that the method is only good at periodic time series. The result of DLinear shows potential effectiveness of linear network for single-step forecasting task and has same speed compared to classic statistical method. Thanks to the introduction of patching

Table 3 The single-step forecasting performance evaluation by MSE and MAE on KPI, Yahoo, NAB, and MSLU datasets. Additionally, the average inference time for each stamp is reported for each method

Method	KPI			Yahoo		
	MSE	MAE	TM (ms)	MSE	MAE	TM (ms)
ARIMA	0.8488	0.8735	0.013	21.3454	1.3756	0.014
Prophet	1.5643	0.8464	0.237	20.9178	0.9897	0.307
GRU	0.2920	0.2098	0.308	20.5051	1.1603	0.300
LSTM	0.2683	0.2026	0.313	20.7485	1.2312	0.312
PAD	0.3883	0.2541	1.024	19.6496	0.9660	0.967
DLinear	0.2058	0.1806	0.115	5.4978	0.3675	0.115
PatchTST	0.1801	**0.1647**	1.530	5.9267	**0.3130**	1.177
iTransformer	0.3389	0.2552	1.192	**4.3630**	0.3294	1.201
PatchLinear	**0.1747**	0.1654	0.207	4.6613	0.3203	0.203
Improvement	3.0%	−0.4%	–	−6.8%	−2.3%	–
Method	**NAB**			**MSLU**		
	MSE	MAE	TM (ms)	MSE	MAE	TM (ms)
ARIMA	1.7561	0.5408	0.024	3.1339	1.1271	0.007
Prophet	2.4999	0.8381	0.428	6.9587	1.7401	0.108
GRU	1.6370	0.5801	0.552	1.7042	0.6662	0.157
LSTM	1.5428	0.5091	0.560	1.9287	0.7227	0.155
PAD	2.8685	0.8314	1.794	1.7980	0.7185	0.512
DLinear	1.5941	0.4806	0.221	1.3342	0.6343	0.061
PatchTST	**1.3170**	**0.4107**	2.708	0.7206	0.3059	0.743
iTransformer	1.4327	0.4295	2.263	1.2592	0.5604	0.613
PatchLinear	1.3878	0.4150	0.394	**0.5807**	**0.2745**	0.106
Improvement	−5.4%	−1.0%	–	24.1%	11.4%	–

The best performance is marked in bold, while the second best performance is underlined

scheme, the forecasting accuracy of our method outperforms DLinear consistently. PatchTST and our method show the best performance across different datasets. Our method achieves the best performance on 3/8 measurements. For the remaining 5/8 measurements, it only shows slightly inferior performance than the much more complicated models PatchTST or iTransformer. When it comes to the forecasting efficiency, our single-step forecasting component shows around *7* times faster speed over PatchTST while achieving similar forecasting accuracy.

4.4 Performance on anomaly detection

For the anomaly detection task, our method is evaluated in comparison to the conventional methods such as DSPOT [38] and SR [36], and methods based on unsupervised networks VAE [22], LSTMAD [30], PAD [5], TFAD [56], and FCVAE [46]. In addition, our method is also compared with RNN-based method LSTMAD [30] and attention-based method AnomalyTransformer [50]. DSPOT detects the anomalies based on the extreme value theory. SR transforms the source series into the frequency domain. The anomalies are reflected as the residuals in the frequency domain. Methods VAE, PAD, and FCVAE are reconstruction-based methods. The anomalies are detected when the discrepancy is significant between the original status and the reconstructed normal status. LSTMAD [30] is comprised by several LSTM layers. Similar as other reconstruction-based method, the anomaly is detected by checking the discrepancy between the forecasted status and the real status. AnomalyTransformer [50] adopts the attention strategies in the time series reconstruction. The anomaly detection is achieved by referring to the reconstructed status as well. TFAD extracts feature vectors for the subsequence (to be considered) and the context subsequence where the subsequence is located in both time and frequency domain. The anomalies are detected based on the distance of their corresponding feature vectors between the subsequence and context subsequence.

The detection performance on four datasets of anomaly detection task is shown in Table 4. The methods are roughly divided into three groups according to their overall performance. Traditional methods DSPOT, and SR show poor performance, and their performance fluctuates significantly across different datasets. These methods fail when the anomalies are insignificant. Methods based on the classic neural networks such as VAE, PAD, LSTMAD and TFAD outperform the traditional methods. However, their performance still fluctuates considerably across different datasets. Compared to PatchLinear, they are either unable to fully capture the correlation between timestamps or are too sensitive to the noises and anomalies. Overall, our method shows

Table 4 The anomaly detection performance evaluation by precision, recall, and F1-score of our method and state-of-the-art approaches on KPI, Yahoo, NAB, and MSLU datasets

Method	KPI			Yahoo		
	Pre.	Rec.	F1	Pre.	Rec.	F1
DSPOT	0.623	0.447	*0.521	0.241	0.458	*0.316
SR	0.647	0.598	*0.622	0.451	0.747	*0.563
VAE	0.725	0.648	*0.685	0.773	0.549	*0.642
PAD	0.839	0.660	*0.739	0.837	0.688	*0.755
LSTMAD	0.786	0.590	0.674	0.413	0.198	0.268
AnomalyTransformer	0.622	0.240	+0.346	0.054	0.020	+0.029
TFAD	0.650	0.714	+0.680	0.883	0.734	+0.802
FCVAE	0.906	0.772	+<u>0.835</u>	0.897	0.792	+**0.842**
PatchLinear	0.858	0.822	**0.840**	0.899	0.788	<u>0.839</u>
Improvement		0.6%			−0.3%	
Method	NAB			MSLU		
	Pre.	Rec.	F1	Pre.	Rec.	F1
DSPOT	0.926	0.457	0.612	0.422	0.175	0.248
SR	0.233	0.067	0.104	0.873	0.932	0.901
VAE	0.904	0.803	0.859	0.965	0.345	0.493
PAD	0.893	0.844	0.868	0.871	0.342	0.473
LSTMAD	0.877	0.510	0.561	0.649	0.765	0.703
AnomalyTransformer	0.891	0.932	+0.911	0.898	0.441	0.591
TFAD	0.265	0.233	+0.248	0.902	0.939	<u>0.920</u>
FCVAE	0.925	0.909	+<u>0.917</u>	0.785	0.742	0.763
PatchLinear	0.901	1.000	**0.948**	0.915	0.982	**0.947**
Improvement		3.4%			2.9%	

The best F1-score is bold and the second best is underlined. * digits are cited from PAD [5], + digits are cited from FCVAE [46]

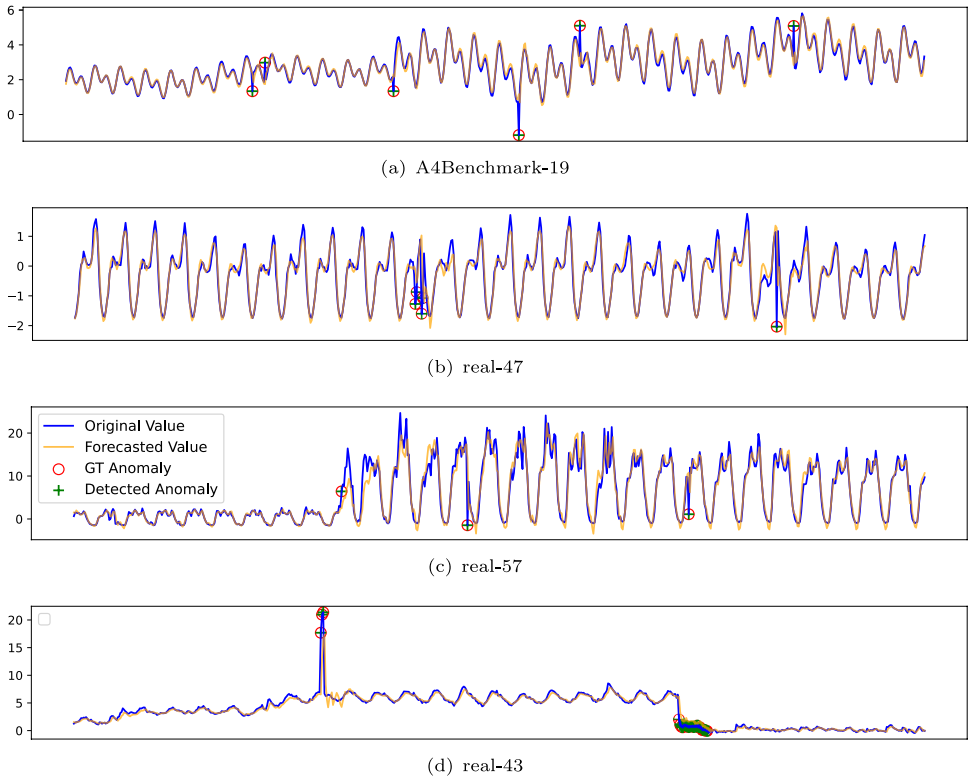
superior performance on most of the datasets than other methods, namely 0.6% on KPI, 3.4% on NAB and 2.9% on MSLU, respectively.

Figure 4 shows the forecasting and anomaly detection results from our method on four time series samples from Yahoo dataset. As shown in the figure, our method shows stable forecasting performance even the time series are under concept drift of different patterns. Moreover, it is able to precisely locate most of the anomalies across different time series.

4.5 Online forecasting and detection

Owing to the concept drift, it is required to update the model periodically. Due to the model complexity or the nature of the model, not all the methods can be deployed online. Only a few methods are able to perform both online single-step forecasting and anomaly detection. In this section, we evaluate our method as an online single-step forecasting and anomaly detection method. PatchTST and PAD are considered as the comparison baseline. Table 5 shows the performance of anomaly detection as well as single-step forecasting performance. The average time cost of processing one timestamp from all the methods are also reported. For each method, the performance from their offline configuration is also reported. The major difference of the online configuration from the offline lies the frequency of the model update. For the online configuration, the model is updated regularly.

As shown in the Table 5, both online PatchLinear and PatchLinear outperform PatchTST and PAD across different datasets considerably. The performance difference between online PatchLinear and PatchLinear is minor. In contrast, the online PatchTST shows considerably inferior performance from PatchTST in most of the cases. This mainly owes to the complex structure of PatchTST model, which requires sufficient data for effectively updating. PAD shows similar performance fluctuation as PatchTST. Since the anomaly detection of all three methods are more or less based on the single-step forecasting. The detection performance trend in Table 5 can be well interpreted by the forecasting performance in Table 5. Moreover, owing to the integration of moving threshold (MT) and distance significance (DS) for anomaly detection, our method outperforms PatchTST considerably even their forecasting performance is similar. In terms of processing efficiency, online runs for all the methods are slower than the offline runs due to the extra time costs in model update. However, all the methods could achieve real-time efficiency. In particular, our method



(a) A4Benchmark-19

(b) real-47

(c) real-57

(d) real-43

Fig. 4 The illustration of the single-step forecasting and the anomaly detection on four time series from Yahoo dataset

Table 5 Performance of PatchLinear, PatchTST, PAD, and their processing efficiency on four datasets. The performance on anomaly detection (F1-score), single-step forecasting (MSE), and efficiency (time cost per timestamp) is reported

Method		KPI			Yahoo		
		F1	MSE	TM (ms)	F1	MSE	TM (ms)
PAD	Offline	0.739	0.3883	1.024	0.755	19.6496	0.967
	Online	0.628	0.4649	1.895	0.722	20.3692	1.796
PatchTST	Offline	0.798	0.1647	1.931	0.531	5.9267	1.769
	Online	0.779	0.2176	4.912	0.466	5.6105	4.439
PatchLinear	Offline	0.840	0.1654	0.392	0.838	4.6613	0.720
	Online	0.845	0.1768	1.096	0.838	4.3629	1.105
Method		NAB			MSLU		
		F1	MSE	TM (ms)	F1	MSE	TM (ms)
PAD	Offline	0.868	2.8685	0.945	0.473	1.7981	0.971
	Online	0.820	1.9389	1.884	0.444	2.0187	2.058
PatchTST	Offline	0.842	1.3170	3.960	0.800	0.7206	1.918
	Online	0.844	1.6233	4.953	0.801	1.0035	4.953
PatchLinear	Offline	0.948	1.3878	1.535	0.947	0.5807	0.576
	Online	0.948	1.3271	1.854	0.945	0.5798	1.015

could process one timestamp within two milliseconds on all the datasets.

4.6 Ablation study

In this section, ablation analysis is carried out to study the contribution of each proposed scheme in the forecasting and anomaly detection. In the ablation study of forecasting, we compare the performance between PatchLinear and Linear model. PatchLinear differs from Linear model in the adopt of patching. As shown in Table 6, the scheme boosts the forecasting performance while involving little computational overhead.

In the ablation study on anomaly detection task, PatchTST is treated as the comparison baseline. The same as the last experiment, the threshold to judge whether a timestamp is anomalous in PathTST is learned on the training data for each dataset. For our method, the basic configuration ("PatchLinear" in Table 7) detects the anomalies based on MAE between the status and the single-step forecasting result. The other innovations over it

Table 6 The ablation study of our method on single-step forecasting task

Method	NAB		Yahoo	
	MSE	TM (ms)	MSE	TM (ms)
Linear	1.4522	6.134	5.6091	1.632
PatchLinear	1.3798	6.152	4.1961	2.755

Table 7 The ablation study of our method on anomaly detection task

Method	KPI		Yahoo	
	F1	TM (ms)	F1	TM (ms)
PatchTST	0.793	1.931	0.531	1.769
PatchLinear	0.794	0.469	0.526	0.702
+ MT	0.812	0.474	0.828	0.679
+ DS	0.827	0.851	0.835	0.854
+ MAD	0.840	0.851	0.839	0.873

are integrated incrementally, such as moving threshold (MT), distance significance (DS), and median absolute deviation (MAD). Table 7 shows the detection performance (F1-score) on KPI and Yahoo datasets from both PathTST and our method under different configurations. The processing time cost per timestamp of each run is also reported. As shown in the table, the performance of our method under basic configuration is similar to PatchTST, which is in line with the observation in the single-step forecasting task. As the proposed schemes are integrated incrementally, the steady performance improvement is observed. Due to the extra time cost required for subsequence matching, the processing time cost increases when DS is integrated. Nevertheless, the time cost is still two times lower than that of PathTST. Moreover, further performance boost is observed when MAD is integrated to address the case that anomaly occurs in the best-matched historical subsequence. The extra time cost introduced by the integration of MAD is negligible.

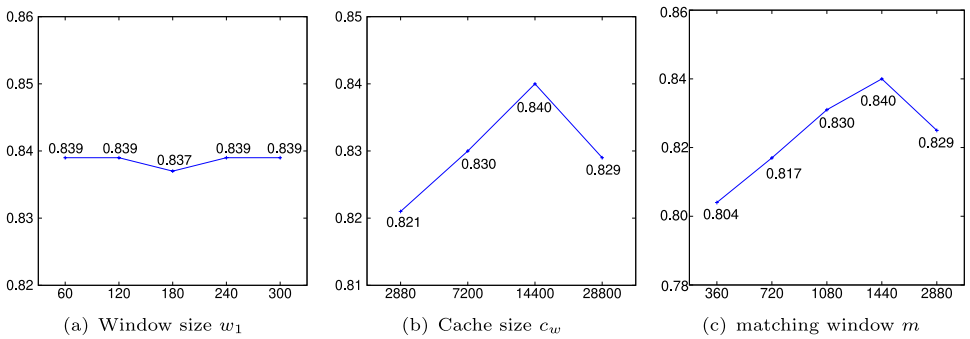
4.7 Parameters sensitivity analysis

There are three hyper-parameters involved in our method, namely w_1, c_w and m. We study the anomaly detection performance trend on dataset KPI when varying one of them while fixing the others. The default values for w_1, c_w, and m are 120, 14, 400, and 1, 440 respectively. In the first test, the sequence window w_1 is varied in the range [60, 360]. As shown in Fig. 5a, the performance of anomaly detection undergoes slight fluctuation. This basically indicates that w_1 only has minor influence on the detection as long as it is in the range [60, 360]. The sensitivity tests for the cache window c_w is shown in Fig. 5b. The best performance is observed when $c_w = 14400$, namely 10 days. This basically reflects that shorter c_w may lack enough historical context to accurately detect anomalies, while longer c_w may introduce excessive outdated information, affecting the performance of anomaly detection. Sensitivity tests for the matching window m is shown in Fig. 5c. The best performance is observed when $m = 1440$, namely 1 day. It indicates that the matching window m should be selected in the same rhythm as the cycle of a time series.

5 Conclusion

We have presented a simple but effective solution for both single-step forecasting and unsupervised anomaly detection on univariate time series. The core component in our solution is a linear neural network. Although simple, it achieves similar performance as the state-of-the-art Transformer-based networks. Moreover, based on the forecasting results, an unsupervised anomaly detection scheme is designed. With the integration with the proposed distance significance, it shows considerably superior performance on all the considered datasets over existing methods. Furthermore, due to the simplicity, our method can be easily deployed as an online method both for forecasting and anomaly detection. And the model size is as small as $10K \sim 30K$ bytes, which is so lightweight that is suitable for various real scenarios.

Although the superior performance is achieved by our method, it is specifically designed for univariate time series. Due to the limitation of the network structure, it is unable to capture the correlation between different variables when facing multivariate time series. In this context, new online detection framework is expected. Moreover, the integration of clustering methods such as SVDD [37], EDCWRN [11], and



(a) Window size w_1 (b) Cache size c_w (c) matching window m

Fig. 5 Performance on KPI dataset with different hyperparameters

SSFCM-FWCW [33] will make the model more effective than the proposed distance significance, which will be our future work.

Acknowledgements We would like to express our sincere thanks to Boden AI Technology Ltd., Ningbo, China, for their support of this work.

Data Availability All source data that support the findings are found from following website https://cmmlab.xmu.edu.cn/resc.html.

References

1. Adari SK, Alla S (2024) Practical use cases and future trends of anomaly detection practical use cases and future trends of anomaly detection. Springer, Berlin, pp 481–509
2. AIOpsChallenge (2017) KPI Anomaly detection competition (2017) KPI anomaly detection. URL http://iops.ai/competition_detail/?competition_id=5
3. Barnett V, Lewis T (1994) Outliers in statistical data outliers in statistical data (3). Wiley, Hoboken
4. Box GE, Jenkins GM (1976). Time series analysis: forecasting and control time series analysis: forecasting and control. Holden-Day, San Francisco
5. Chen R, Shi G, Zhao W, Liang C (2021) A joint model for IT operation series prediction and anomaly detection a joint model for IT operation series prediction and anomaly detection. Neurocomputing 448:130–139. https://doi.org/10.1016/j.neucom.2021.03.062
6. Chen S, Chung N (2006) Forecasting enrollments using high-order fuzzy time series and genetic algorithms. Int J Intell Syst 21:485–501. https://doi.org/10.1002/int.20145
7. Cho K, van Merriënboer B, Gulçehre Ç, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the EMNLP, pp 1724–1734, https://hal.science/hal-01433235/
8. Cirstea RG, Guo C, Yang B, Kieu T, Dong X, Pan S (2021) Triformer: triangular, variable-specific attentions for long sequence multivariate time series forecasting triformer: triangular, variable-specific attentions for long sequence multivariate time series forecasting. In: Proceedings of the IJCAI, pp 1–18, https://doi.org/10.48550/arXiv.2204.13767
9. Connor JT, Martin RD, Atlas LE (1994) Recurrent neural networks and robust time series prediction recurrent neural networks and robust time series prediction. IEEE Trans Neural Netw 240–254, https://doi.org/10.1109/72.279188
10. Dang Y, Lin Q, Huang P (2019) AIOps: Real-world challenges and research innovations AIOps: real-world challenges and research innovations. In: Proceedings of the ICSE, pp 4–5, https://doi.org/10.1109/ICSE-Companion.2019.00023
11. Golzari Oskouei A, Balafar MA, Motamed C (2023) EEDC-WRN: efficient deep clustering with the weight of representations and the help of neighbors EDCWRN: efficient deep clustering with the weight of representations and the help of neighbors. Appl Intell 53:5845–5867. https://doi.org/10.1007/s10489-022-03895-5
12. Gooijer JGD, Hyndman RJ (2006) 25 years of time series forecasting. Int J Forecast 22:443–473. https://doi.org/10.1016/j.ijforecast.2006.01.001
13. Han Z, Zhao J, Leung H, Ma KF, Wang W (2021) A review of deep learning models for time series prediction a review of deep learning models for time series prediction. IEEE Sensors Journal 7833 – 7848, https://doi.org/10.1109/JSEN.2019.2923982
14. Harvey AC (1990) Forecasting, Structural Time Series Models and the Kalman Filter Forecasting. Cambridge University Press, Cambridge
15. Hawkular (2014). Hawkular for monitoring services: metrics, alerting, inventory, application performance management. https://github.com/hawkular
16. Hochreiter S, Jürgen S (1997). Long short-term memory. Neural Comput 1735–1780, https://doi.org/10.1162/neco.1997.9.8.17359
17. Holland JH (1992) Genetic algorithms. Sci Am 267:66–73 (http://www.jstor.org/stable/24939139)
18. Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T (2018) Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Proceedings of the SIGKDD, pp 387–395, https://doi.org/10.1145/3219819.3219845
19. Kalekar PS (2004) Time series forecasting using holt-winters exponential smoothing. Kanwal Rekhi School of Information Technology, pp 1–13
20. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN, pp 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
21. Kim T, Kim J, Tae Y, Park C, Choi JH, Choo J (2021) Reversible instance normalization for accurate time-series forecasting against distribution shift. In: Proceedings of the ICLR. https://openreview.net/forum?id=cGDAkQo1C0p
22. Kingma DP, Welling M (2013). Auto-encoding variational bayes auto-encoding variational bayes. arXiv preprint arXiv:1312.6114. https://doi.org/10.48550/arXiv.1312.6114
23. Lavin A, Ahmad S (2015) Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In: Proceedings of the ICMLA, pp 38–44. https://doi.org/10.1109/ICMLA.2015.141
24. LeCun Y (1987) Connexionist learning models connexionist learning models . ParisUniversite Pierre et Marie Curie
25. Leys C, Ley C, Klein O, Bernard P, Licata L (2013) Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. J Exp Soc Psychol, pp 764–766. https://doi.org/10.1016/j.jesp.2013.03.013
26. Li D, Chen D, Jin B, Shi L, Goh J, Ng SK (2019) MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. In: Proceedings of the ICANN, pp 703–716. https://doi.org/10.1007/978-3-030-30490-4_56
27. Li Z, Qi S, Li Y, Xu Z (2023) Revisiting long-term time series forecasting: an investigation on linear mapping. arXiv preprint arXiv:2305.10721. https://doi.org/10.48550/arXiv.2305.10721
28. Liu S, Liu S, Yu H, Liao C, Li J, Lin W, Dustdar S (2022) Pyraformer: low-complexity pyramidal attention for long-range time series modeling and forecasting. Proceedings of ICLR, pp 1–12. https://doi.org/10.34726/2945
29. Liu Y, Hu T, Zhang H, Wu H, Wang S, Ma L, Long M (2024) iTransformer: inverted transformers are effective for time series forecasting. In: Proceedings of the ICLR. https://openreview.net/forum?id=JePfAI8fah
30. Malhotra P, Vig L, Shroff G, Agarwal P (2015) Long short term memory networks for anomaly detection in time series. Esann, 89
31. Mueen A, Keogh E, Zhu Q, Cash S, Westover B (2009) Exact discovery of time series motifs. In: Proceedings of the ICDM, pp 473-484. https://doi.org/10.1137/1.9781611972795.41
32. Nie Y, Nguyen NH, Sinthong P, Kalagnanam J (2023) A time series is worth 64 words: long-term forecasting with transformers.

In: Proceedings of ICLR. https://doi.org/10.48550/arXiv.2211.14730
33. Oskouei AG, Samadi N, Tanha J, Bouyer A (2024) SSFCM-FWCW: semi-supervised fuzzy C-means method based on feature-weight and cluster-weight learning. Softw Impacts 21:100678. https://doi.org/10.1016/j.simpa.2024.100678
34. Park D, Hoshi Y, Kemp CC (2018) A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. IEEE Robot Autom Lett 3:1544–1551. https://doi.org/10.1109/TMC.2019.2944829
35. Qin Y, Song D, Cheng H, Cheng W, Jiang G, Cottrell GW (2017) A dual-stage attention-based recurrent neural network for time series prediction. In: Proceedings of the IJCAI, pp 2627–2633. https://doi.org/10.24963/ijcai.2017/366
36. Ren H, Zhang Q, Xu B, Wang Y, Yi C, Huang C, Tong J (2019) Time-series anomaly detection service at microsoft. In: Proceedings of the SIGKDD, pp 3009–3017. https://doi.org/10.1145/3292500.3330680
37. Ruff L, Vandermeulen R, Goernitz N, Deecke L, Siddiqui SA, Binder A, Kloft M (2018) Deep one-class classification. In: Proceedings of the ICML, pp 4393–4402. https://proceedings.mlr.press/v80/ruff18a.html
38. Siffer A, Fouque PA, Termier A, Largouet C (2017) Anomaly detection in streams with extreme value theory. In: Proceedings of the SIGKDD, pp 1067–1075. https://doi.org/10.1145/3097983.3098144
39. Singh P (2017) A brief review of modeling approaches based on fuzzy time series. Int J Mach Learn Cybern 8:397–420. https://doi.org/10.1007/s13042-015-0332-y
40. Singh P (2020) A novel hybrid time series forecasting model based on neutrosophic-PSO approach. Int J Mach Learn Cybern 11:1643–1658. https://doi.org/10.1007/s13042-020-01064-z
41. Song H, Rajan D, Thiagarajan J, Spanias A (2018) Attend and diagnose: clinical time series analysis using attention models. In: Proceedings of the AAAI, pp 4091–4098. https://doi.org/10.1609/aaai.v32i1.11635
42. Su Y, Zhao Y, Niu C, Liu R, Sun W, Pei D (2019) Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the SIGKDD, pp 2828–2837. https://doi.org/10.1145/3292500.3330672
43. Taylor SJ, Letham B (2018) Forecasting at scale. Am Stat, pp 37–45. https://doi.org/10.1080/00031305.2017.1380080
44. Tuli S, Casale G, Jennings NR (2022) TranAD: deep transformer networks for anomaly detection in multivariate time series data. In: Proceedings of the VLDB endowment, vol 15, pp 1201–1214, https://doi.org/10.14778/3514061.3514067
45. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Polosukhin I (2017) Attention is all you need. Advances in neural information processing systems, pp 6000–6010. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
46. Wang Z, Pei C, Ma M, Wang X, Li Z, Pei D et al (2024) Revisiting VAE for unsupervised time series anomaly detection: a frequency perspective. In: Proceedings of the WWW DOI 10(1145/3589334):36457109
47. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn, pp 69–101. https://doi.org/10.1023/A:1018046501280
48. Wu, H. , Xu, J. , Wang, J. Long, M. 2021. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. Advances in neural information processing systems, pp 22419–22430. https://doi.org/10.5555/3540261.3541978
49. Xu H, Chen W, Zhao N, Li Z, Bu J, Li Z, others (2018) Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In: Proceedings of the WWW, pp 187–196. https://doi.org/10.1145/3178876.3185996
50. Xu J, Wu H, Wang J, Long M (2022) Anomaly transformer: time series anomaly detection with association discrepancy. In: Proceedings of the ICLR. https://openreview.net/forum?id=LzQQ89U1qm_
51. Xue J, Yan F, Chen LY, Scherer T, Smirni E (2015) PRACTISE: robust prediction of data center time series PRACTISE: robust prediction of data center time series. In: Proceedings of the CNSM, pp 126–134. https://doi.org/10.1109/CNSM.2015.7367348
52. YahooLabs S (2015) A labeled anomaly detection dataset, version 1.0 (2015). A labeled anomaly detection dataset, version 1.0 (2015). https://webscope.sandbox.yahoo.com/catalog.php?datatype=s%26did=70
53. Yeh C CM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Keogh E (2016) Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and Shapelets matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and Shapelets. In: Proceedings of the ICDM, pp 1317–1322. https://doi.org/10.1109/ICDM.2016.0179
54. Zameer A, Arshad J, Khan A, Raja MAZ (2017) Intelligent and robust prediction of short term wind power using genetic programming based ensemble of neural networks intelligent and robust prediction of short term wind power using genetic programming based ensemble of neural networks. Energy Convers Manage. https://doi.org/10.1016/j.enconman.2016.12.032
55. Zeng A, Chen M, Zhang L, Xu Q (2023) Are transformers effective for time series forecasting? Are transformers effective for time series forecasting?. In: Proceedings of the AAAI, pp 11121–11128. https://doi.org/10.1609/aaai.v37i9.26317
56. Zhang C, Zhou T, Wen Q, Sun L (2022) TFAD: a decomposition time series anomaly detection architecture with time-frequency analysis TFAD: a decomposition time series anomaly detection architecture with time-frequency analysis. In: Proceedings of the CIKM, pp 2497–2507. https://doi.org/10.1145/3511808.3557470
57. Zhang Z, Li W, Ding W, Zhang L, Lu Q, Hu P, Lu S (2023) STAD-GAN: unsupervised anomaly detection on multivariate time series with self-training generative adversarial networks STAD-GAN: unsupervised anomaly detection on multivariate time series with self-training generative adversarial networks. ACM transactions on knowledge discovery from data, pp 1–18. https://doi.org/10.1145/35727809
58. Zhou B, Liu S, Hooi B, Cheng X, Ye J (2019) BeatGAN: anomalous rhythm detection using adversarially generated time series. BeatGAN: anomalous rhythm detection using adversarially generated time series. In: Proceedings of the IJCAI, pp 4433–4439. https://dl.acm.org/doi/abs/10.5555/3367471.3367658
59. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021). Informer: beyond efficient transformer for long sequence time-series forecasting informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI, pp 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325
60. Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R (2022) FEDformer: frequency enhanced decomposed transformer for long-term series forecasting FEDformer: frequency enhanced decomposed transformer for long-term series forecasting. In: Proceedings of ICML, pp 1–19. https://proceedings.mlr.press/v162/zhou22g.html
61. Zhu Y, Zimmerman Z, Senobari NS, Yeh CCM, Funning G, Mueen A, Keogh E (2016) Matrix Profile II: exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins matrix profile II: exploiting a novel algorithm and GPUs to break the one hundred million barrier for

time series motifs and joins. In: Proceedings of the ICDM, pp 739–748. https://doi.org/10.1109/ICDM.2016.0085