TimeAutoAD: Autonomous Anomaly Detection with Self-supervised Contrastive Loss for Multivariate Time Series

Yang Jiao, Kai Yang, *Senior Member, IEEE,* Dongjing Song, *Member, IEEE,* Dacheng Tao, *Fellow, IEEE*

***Abstract*—Multivariate time series (MTS) data are becoming increasingly ubiquitous in networked systems, e.g., IoT systems and 5G networks. Anomaly detection in MTS refers to identifying time series which exhibit different behaviors from normal status. Building such a system, however, is challenging due to a few reasons: i) labels for anomaly cases are usually unavailable or very rare; ii) most existing approaches rely on manual model-design and hyperparameter tuning, which may cost a huge amount of labor effort. To this end, we propose an autonomous anomaly detection technique for multivariate time series data (TimeAutoAD) based on a novel self-supervised contrastive loss. Specifically, we first present an automatic anomaly detection pipeline to optimize the model configuration and hyperparameters automatically. Next, we introduce three different strategies to augment the training data for generating pseudo negative time series and employ a self-supervised contrastive loss to distinguish the original time series and the generated time series. In this way, the representation learning capability of TimeAutoAD can be greatly enhanced and the anomaly detection performance can thus be improved. Extensive empirical studies on real-world datasets demonstrate that the proposed TimeAutoAD not only outperforms state-of-the-art anomaly detection approaches but also exhibits robustness when training data are contaminated.**

***Index Terms*—Multivariate time series, anomaly detection, Automatic Machine Learning (AutoML), self-supervised learning, contrastive loss.**

I. Introduction

The past decade has witnessed a rising proliferation in Multivariate Time Series (MTS) data, along with a plethora of applications in domains as diverse as IoT data analysis [1], medical informatics [2], and network security [3]. Given the huge amount of MTS data collected from these systems, it is important to detect anomalous time series which exhibit different behaviors from normal status [4, 5]. For instance, the anomalies in electrocardiogram represent heart arrhythmia. Detecting these anomalies is beneficial to the diagnosis of heart disease. Moreover, for a larger number of time series data, it is time-consuming to manually construct an anomaly detection pipeline and optimize hyperparameters. Thus, it is meaningful to design an automatic machine learning model for time series anomaly detection.

Tremendous efforts have been devoted to anomaly detection, existing anomaly detection approaches can be divided into four categories, i.e., classification based, distance based, clustering based, and statistical anomaly detection methods [6, 7]. Classification based anomaly detection methods aim at learning a classifier from a set of labeled samples and can be grouped into one-class [8–10] and multi-class [11, 12] classification based methods depending on the label availability in training phase. One-class SVM [8] assumes that all training data belongs to the same class and aims at finding a maximum margin hyperplane in the feature space to separate the normal data from the anomalies. Toupas et al. [12] implement multi-class classification based anomaly detection in intrusion detection systems utilizing a neural network model. Distance based anomaly detection methods assume that anomalous data locate far from normal data and require a distance measure for decision-making. Ramaswamy et al. [13] propose to detect anomalies based on the distance of a sample from its k^{th} nearest neighbor. Jin et al. [14] extract feature from samples for Mahalanobis distance calculation and detect anomalies based on a predefined threshold. Clustering based anomaly detection methods aim at grouping similar samples into clusters. k-Means [15], k-Medoids [16] and Gaussian Mixture Models [17] are well-known clustering based anomaly detection methods. Furthermore, deep autoencoder based methods [18, 19] have been proposed recently, which jointly perform dimensionality reduction and clustering analysis in the end-to-end manner. Statistical anomaly detection methods assume that normal data appear in the high probability area of the stochastic model, while anomalies appear in the low probability area. For example, Wang et al. [20] propose a statistical anomaly detection method based on Tukey and Relative Entropy statistics, which analyzes the monitoring data over historical periods to make decisions.

Automated machine learning (AutoML) has recently received intensive attention. We concentrate on the automatic pipeline configuration problem [21], i.e., joint module selection and hyperparameter optimization. There are two main challenges against automatic pipeline configuration: 1) the black-box nature of optimization objective; 2) the tight coupling between AutoML pipeline configuration and hyperparameter optimization. Auto-WEKA [22] applies a general purpose framework, i.e., sequential purpose model based algorithm configuration to find optimal machine learning pipelines. Liu et al. [21] propose an ADMM based method, which optimizes the AutoML pipeline via a primal-dual decomposition approach. Nevertheless, to our best knowledge, little work has addressed the design of automatic anomaly detection pipeline

Yang Jiao and Kai Yang are with the Department of Computer Science and Technology, Tongji University, Shanghai, China

Dongjing Song is with Deparment of Computer Science, University of Connecticut

Dacheng Tao is with the JD Explore Academy in JD.com, China

Manuscript received April xxx; revised August xxx.

configuration.

Despite many previous works have been developed for anomaly detection in multivariate time series, there are still a few challenges preventing them from building an effective model: i) the design of an anomaly detection model and the tuning for hyperparameters often employ a trial and error procedure which are time consuming and could cost a substantial amount of labor effort; ii) the labels for anomaly cases are usually unavailable or very rare. Most existing approaches assume all the training data are normal and train their models based on normal samples. Nevertheless, the use of abnormal samples is critical in the training procedure of anomaly detection models. For example, the performance of semi-supervised anomaly detection methods has been proven to be superior to unsupervised methods since such methods optimize the anomaly detection capability according to a few labeled anomalies [23, 24].

To address the aforementioned challenges, we propose an autonomous unsupervised anomaly detection model with a novel self-supervised contrastive loss, a.k.a. TimeAutoAD, to perform anomaly detection in multivariate time series data. Specifically, TimeAutoAD differs from traditional time series anomaly detection approaches in three aspects. First, the anomaly detection pipeline configuration and hyperparameter optimization are carried out automatically in TimeAutoAD. Second, several negative sample generation approaches are proposed to generate negative samples for contrastive training. Finally, a novel self-supervised contrastive loss is proposed to distinguish the original samples and the generated negative samples. In this way, the representation learning capability of TimeAutoAD is greatly enhanced, leading to superior anomaly detection performance [25–27]. We conduct extensive experiments on several multivariate real-world anomaly datasets and numerous datasets in UCR and UEA archives. Our experiments demonstrate that the proposed TimeAutoAD outperforms state-of-the-art anomaly detection algorithms, and the proposed TimeAutoAD is robust against the contaminated training data.

Our contributions can be summarized as:

(1) TimeAutoAD has been proposed, which can automatically configure anomaly detection pipeline and optimize its hyperparameters. As far as we are aware of, this is the first work that consider joint automatic ML pipeline configuration, hyperparameter optimization, and negative sample generation for time series anomaly detection.

(2) We have designed a novel self-supervised contrastive loss which can be used to effectively enhance the anomaly detection performance. In addition, several time series negative sample generation methods have been proposed for contrastive training.

(3) Extensive experiments have been conducted on numerous datasets in UCR and UEA archives. It is seen that TimeAutoAD achieves best anomaly detection performance over most $> 90\%$ datasets.

The remainder of this paper is organized as follows. Section II introduces the prior art on unsupervised anomaly detection methods and Automatic Machine Learning. Section III discusses the design of TimeAutoAD framework and proposes the self-supervised contrastive loss. Section IV provides extensive experiment results to demonstrate the superior performance of the proposed TimeAutoAD. Finally, the conclusion of this paper is given in Section V.

II. Related Work

A. Unsupervised Anomaly Detection

Tremendous efforts have been taken for anomaly detection, this paper focuses on unsupervised anomaly detection (i.e., no labeled data in trainging phase). Unsupervised anomaly detection methods are widely applied in real-world applications due to their advantage of not requiring any labeled data. Existing techniques for unsupervised anomaly detection can be grouped into three categories, i.e., one-class classification methods, clustering-based methods, and reconstruction-based methods.

For one-class classification methods, One-Class SVM [8] aims at finding a maximum margin hyperplane in the feature space to separate the normal data from the anomalies. Based on One-Class SVM, Support Vector Data Description (SVDD) [9] uses a hypersphere instead of the hyperplane to enclose the majority of the data in feature space. Both of these methods, however, cannot work well with high-dimensional data due to the curse of dimensionality. As a remedy, Deep SVDD [10] is proposed to train a neural network while minimizing the volume of a hypersphere. Recently, a temporal one-class classification model named THOC is proposed [28], which utilized multiple hyperspheres obtained with a hierarchical clustering process for anomaly detection.

Clustering based methods can be further categorized as partitioning based methods, density based methods and grid based methods. Partitioning-based methods partition data into a few cluster. *CD-trees* [29] is proposed to partition data into clusters efficiently, the data in the sparse cluster are deemed as anomalies. Density-based methods perform anomaly detection by estimating the density of samples and the samples located in low-density area are deemed as anomalies. Breunig et al. [30] propose LOF, which assumes that the local density of anomalies will be lower than that of its nearest neighbors. Grid-based methods perform clustering on a grid structure. Zhong et al. [31] propose an algorithm based cell partition which can generate cluster centers automatically. The generated clusters can further be labeled as normal or attack clusters and the samples in the attack clusters are deemed as anomalies.

Reconstruction based methods determine whether an input time series is normal or abnormal based on the reconstruction error. Principal Component Analysis (PCA) [32] is a conventional reconstruction-based method, which reconstructs data based on linear transformation. Different from PCA, Auto-encoder [33] is more flexible, which can reconstruct data based on both of linear and non-linear transformation. BeatGAN [25] uses auto-encoder as a generator to reconstruct samples and utilizes the Generative Adversarial Networks (GANs) to regularize the reconstruction error. However, reconstruction-based methods also have many shortcomings since such methods do not consider the latent space data distribution. To address this problem, DAGMM [34] combines a deep auto-encoder and the Gaussian Mixture Model (GMM) to perform anomaly detection. Nevertheless, DAGMM is not tailored for anomaly detection in MTS.

Different from the previous anomaly detection approaches, we propose TimeAutoAD, which can automatically configure an unsupervised time series anomaly detection pipeline and simultaneously optimize its hyperparameters. Moreover, a novel self-supervised contrastive loss is proposed to enhance the representation learning ability of the proposed model.

B. *Automatic Machine Learning*

Automatic Machine Learning (AutoML) aims to automate the time-consuming model development process and has received a significant amount of research interest recently. Previous works about AutoML mostly focus on the domains of computer vision and natural language processing, including object detection [35, 36], semantic segmentation [37, 38], translation [39], and sequence labeling [40]. However, AutoML for time series learning is an underappreciated topic so far and the existing works mainly focus on supervised learning tasks. Ukil et al. [41] propose an AutoML pipeline for automatic feature extraction and feature selection for time series classification. Kuppevelt et al. [42] develop an AutoML framework for supervised time series classification, which involves both neural architecture search and hyperparameter optimization. Olsavszky et al. [43] propose a framework called AutoTS, which performs time series forecasting of multiple diseases. Li et al. [44] propose AutoOD, which is an automated outlier detection framework and aims at searching for the optimal neural network model within a predefined search space. Nevertheless, to our best knowledge, no previous work has addressed time series anomaly detection based on AutoML. To this end, we design an AutoML framework to perform time series anomaly detection, which can automatically configure an anomaly detection pipeline and optimize its hyperparameters simultaneously.

III. Proposed Method

Let $\boldsymbol{X} = \{\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2, \cdots \underline{\boldsymbol{x}}_N\}$ denote a set of N multivariate time series. $\underline{\boldsymbol{x}}_i \in \mathbb{R}^{D\times T_i}$ in $\boldsymbol{X}$ represents the i^{th} sample with D dimensions and T_i length. As a special case, D equals to 1 for univariate time series.

A. *TimeAutoAD Architecture*

For a copious amount of multivariate time series data, it is time-consuming to manually construct an anomaly detection pipeline and optimize its hyperparameters. To address this challenge, the proposed TimeAutoAD framework can automatically configure an anomaly detection pipeline with an array of functional modules, each of these modules is associated with a set of hyperparameters. We assume there are a total of M modules and there are Q_i options for the i^{th} functional module. Let $\underline{\boldsymbol{k}}_i \in \{0, 1\}^{Q_i}$ denotes an indicating vector for i^{th} module, with the constraint $1^\top \underline{\boldsymbol{k}}_i = \sum_{j=1}^{Q_i} k_{i,j} = 1$ ensuring that only a single option is chosen for each module. Let $\underline{\boldsymbol{\theta}}_{i,j}^C \in C_{i,j}$ and $\underline{\boldsymbol{\theta}}_{i,j}^D \in D_{i,j}$ respectively be the continuous and discrete hyperparameters of j^{th} option in i^{th} module (constrained to the set $C_{i,j}$ and $D_{i,j}$). Let $\{\Theta^C, \Theta^D\}$ denote the set of variables to optimize, i.e., $\Theta^C = \{\underline{\boldsymbol{\theta}}_{i,j}^C, \forall i \in [M], j \in [Q_i]\}$, $\Theta^D = \{\underline{\boldsymbol{\theta}}_{i,j}^D, \forall i \in [M], j \in [Q_i]\}$, and $\mathrm{K} = \{\underline{\boldsymbol{k}}_1, \ldots, \underline{\boldsymbol{k}}_M\}$. We further let $f(\mathrm{K}, \{\Theta^C, \Theta^D\})$ denote the corresponding objective function value and we use Area Under the Receiver Operating Curve (AUC) as the objective function for anomaly detection. The overall optimization problem can be expressed as follows:

$$\max_{\mathrm{K}, \Theta^C, \Theta^D} f(\mathrm{K}, \{\Theta^C, \Theta^D\})$$
$$\text{subject to} \begin{cases} \underline{\boldsymbol{k}}_i \in \{0, 1\}^{Q_i}, 1^\top \underline{\boldsymbol{k}}_i = 1, \forall i \in [M], \\ \underline{\boldsymbol{\theta}}_{i,j}^C \in C_{i,j}, \underline{\boldsymbol{\theta}}_{i,j}^D \in D_{i,j}, \forall i \in [M], j \in [Q_i]. \end{cases} \tag{1}$$

We solve the above problem (1) based on alternating optimization, which is presented in Algorithm 1. The rationale behind the advantage of alternating optimization is that it decomposes the AutoML problem into sub-problems with small number of variables. And this is crucial for the optimization of $f(\mathrm{K}, \{\Theta^C, \Theta^D\})$ whose convergence is strongly dependent on the number of variables. Indeed, for n variables, the number of evaluations needed for critical point convergence is typically $O(n \sim n^3)$ [45].

1) Anomaly Detection Pipeline Configuration: We first assume that the hyperparameters $\{\Theta^C, \Theta^D\}$ are fixed during the pipeline configuration. We aim at selecting the better module option K to optimize objective function $f(\mathrm{K}, \{\Theta^C, \Theta^D\})$, we can delineate it as a $\mathbf{K}-\mathbf{max}$ problem:

$$\mathrm{K}^{(t+1)} = \max_{\mathrm{K}} f(\mathrm{K}, \{\Theta^{C(t)}, \Theta^{D(t)}\}) + \chi_K(\mathrm{K}),$$
$$\chi_K(\mathrm{K}) = \begin{cases} 0, & \text{if } \mathrm{K} \in K \\ -\infty, & \text{else} \end{cases}, \tag{2}$$

where K is the feasible set, i.e., $K = \{\mathrm{K} : \mathrm{K} = \{\underline{\boldsymbol{k}}_i\}, \underline{\boldsymbol{k}}_i \in \{0, 1\}^{Q_i}, 1^\top \underline{\boldsymbol{k}}_i = 1, \forall i \in [M]\}$ and $\chi_K(\mathrm{K})$ is a penalty term that makes sure K belongs to the feasible region.

Problem (2) can be interpreted as a combinatorial multi-armed bandit problem as the selection of the optimal M arms (in this case, modules) from $\sum_{i=1}^{M} Q_i$ on bandit feedback. We propose a customized Thompson Sampling to effectively solve this multi-armed bandit problem. The detailed information about the proposed customized Thompson Sampling is presented in Algorithm 2.

There are nine modules in our anomaly detection pipeline (as elucidated in Fig. 1), which are all necessary for anomaly detection. Specifically, our anomaly detection pipeline consists of three key parts, i.e., auto representation learning, auto anomaly score calculation and auto negative sample generation. Some modules (i.e., encoder, decoder, attention layer, similarity selection, data augmentation and auxiliary classification network modules) are used for auto representation learning. Some modules (i.e., EM estimator and estimation network modules) are used for auto anomaly score calculation. And negative sample generation module is used to generate negative samples for contrastive training.

Similar to DAGMM [34], we also detect anomalies based on the anomaly score that is calculated according to the latent

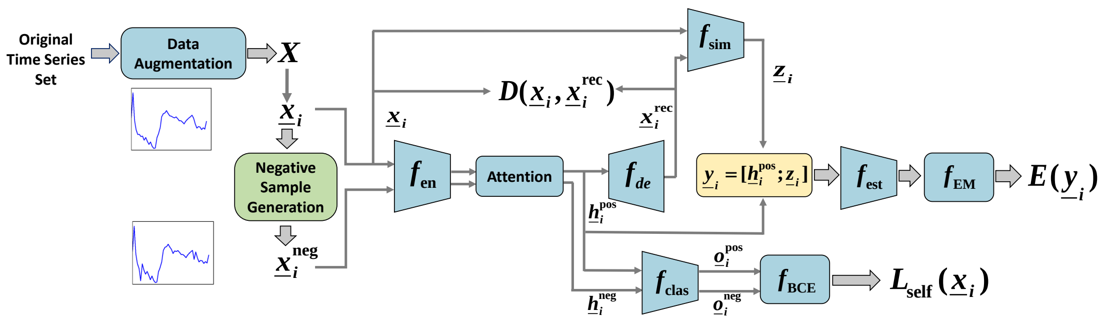

Fig. 1. The anomaly detection pipeline of TimeAutoAD. There are totally nine modules forming this pipeline, namely data augmentation, negative sample generation, encoder f_{en}, attention mechanism, decoder f_{de}, similarity measurement selection f_{sim}, estimation network f_{est} , EM estimator f_{EM}, auxiliary classification network f_{clas}. And f_{BCE} represents the binary cross entropy computation, $D(\boldsymbol{x}_i, \boldsymbol{x}_i^{\text{rec}})$, $E(\boldsymbol{y}_i)$ and $L_{\text{self}}(\boldsymbol{x}_i)$ represent the reconstruction error, sample energy and self-supervised contrastive loss of input sample $\boldsymbol{x}_i$, respectively.

Algorithm 1: Operator splitting from alternating optimization to solve problem (1)

K − max: $$\mathrm{K}^{(t+1)} = \max_{\mathrm{K}} f(\mathrm{K}, \{\Theta^{C(t)}, \Theta^{D(t)}\}) + \chi_K(\mathrm{K}),$$

Θ − max: $$\{\Theta^{C(t+1)}, \Theta^{D(t+1)}\} = \max_{\Theta^C, \Theta^D} f(\mathrm{K}^{(t)}, \{\Theta^C, \Theta^D\}) + \chi_C(\Theta^C) + \chi_D(\Theta^D),$$

where (t) represents the iteration index.

space representation of the input data. Firstly, data augmentation module is utilized to increase the sample diversity, which will be discussed in detail in Appendix C. And the encoder, attention mechanism, decoder and similarity measurement selection modules are utilized together to generate the low-dimensional representation of the input time series. Given an input time series $\boldsymbol{x}_i$, we can first acquire the encoder latent states $\boldsymbol{h}_i$ as:

$$\boldsymbol{h}_i = f_{\text{en}}(\boldsymbol{x}_i), \tag{3}$$

where f_{en} refers to an encoder with an optional attention mechanism. The use of attention layer could lead to better latent space representation and we evaluate the effect of attention layer by comparing the performance of the anomaly detection pipeline with and without attention layer, which is presented in Appendix D. Then, the reconstructed time series $\boldsymbol{x}_i^{\text{rec}}$ and the reconstruction error can be generated as follows:

$$\begin{aligned} \boldsymbol{x}_i^{\text{rec}} &= f_{\text{de}}(\boldsymbol{h}_i), \\ \boldsymbol{z}_i &= f_{\text{sim}}(\boldsymbol{x}_i, \boldsymbol{x}_i^{\text{rec}}), \end{aligned} \tag{4}$$

where f_{de} and f_{sim} represent the decoder and the similarity measurement function, respectively. Specifically, there are three options for the encoder and decoder, namely, Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU). And f_{sim} characterizes the level of similarity between the original time series and the reconstructed one. Three possible similarity measurement functions are considered in this paper, i.e., relative Euclidean distance, Cosine similarity, or concatenation of both. Finally, we can get the low-dimensional representation of the input time series, which is a concatenation of encoder latent states $\boldsymbol{h}_i$ and reconstruction error $\boldsymbol{z}_i$, as shown below:

$$\boldsymbol{y}_i = [\boldsymbol{h}_i; \boldsymbol{z}_i], \tag{5}$$

Once the latent space representation $\boldsymbol{y}_i$ of the input time series is acquired, Gaussian Mixture Model (GMM) is used to fit the distribution of the latent space representation. The estimation network and EM estimator are utilized to estimate the mixture probability, mean and convariance of GMM. Assuming there are H mixture components in the GMM model, the mixture probability, mean, covariance for component h in the GMM module can be respectively expressed as $\phi_h, \boldsymbol{\mu}_h, \sum_h$ and can be calculated as:

$$\begin{aligned} &\boldsymbol{\gamma}_i = f_{\text{est}}(\boldsymbol{y}_i), \forall i \in [N], \\ &\phi_h = \textstyle\sum_{i=1}^{N} \frac{\gamma_{i,h}}{N}, \forall h \in [H], \\ &\boldsymbol{\mu}_h, \textstyle\sum_h = f_{\text{EM}}(\{\boldsymbol{y}_i, \boldsymbol{\gamma}_{i,h}\}_{i=1}^{N}), \forall h \in [H], \end{aligned} \tag{6}$$

where N is the number of input samples, f_{est} is the estimation network which is a multi-layer feed-forward neural network and $\boldsymbol{\gamma}_i \in \mathbb{R}^H$ is the mixture-component membership prediction vector. f_{EM} is the EM estimator which can estimate the mean and convariance of GMM via the EM algorithm. The h^{th} entry of this vector represents the probability that $\boldsymbol{y}_i$ belongs to the h^{th} mixture component.

GMM is employed in this framework for two reasons. First, GMM is a flexible and powerful model that has been proved to be capable of approximating any continuous distribution arbitrarily well under mild assumptions. Second, once we obtain a GMM model using the training data, we can calculate the distance between an input time series and the centroids of GMM in the latent space, which are proportional to the sample energy function. Therefore, the GMM model, in combination with the sample energy function, help to characterize the level of abnormality of an input time series. The time series that is far away from the centroids of the GMM in the latent space will be deemed as an anomaly. And it is worth noticing that

TimeAutoAD may suffer from the *singularity* problem as in GMM. In this case, the training algorithm may converge to a trivial solution if the covariance matrix is singular. We prevent this singularity problem by adding $1e-6$ to the diagonal entries of the covariance matrices.

We next introduce three crucial functions in our anomaly detection process: 1) reconstruction error $D(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_i^{\text{rec}})$, 2) the proposed self-supervised contrastive loss $L_{\text{self}}(\underline{\boldsymbol{x}}_i)$, 3) sample energy $E(\underline{\boldsymbol{y}}_i)$. For reconstruction error $D(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_i^{\text{rec}})$, it represents the difference between the input time series $\underline{\boldsymbol{x}}_i$ and the reconstructed time series, which can be expressed as:

$$D(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_i^{\text{rec}}) = ||\underline{\boldsymbol{x}}_i - \underline{\boldsymbol{x}}_i^{\text{rec}}||_2^2. \tag{7}$$

For the proposed self-supervised contrastive loss $L_{\text{self}}(\underline{\boldsymbol{x}}_i)$, it is related to the negative sample generation and auxiliary classification network modules, which will be discussed in Section III-B. Sample energy $E(\underline{\boldsymbol{y}}_i)$ of the input time series $\underline{\boldsymbol{x}}_i$ is used to characterize the level of anomaly. $E(\underline{\boldsymbol{y}}_i)$ can be calculated as follows:

$$E(\underline{\boldsymbol{y}}_i) = -\log\Big(\sum_{h=1}^{H} \phi_h \cdot \frac{\exp(-\frac{1}{2}(\underline{\boldsymbol{y}}_i - \underline{\boldsymbol{\mu}}_h)^\top \sum_h^{-1} (\underline{\boldsymbol{y}}_i - \underline{\boldsymbol{\mu}}_h))}{\sqrt{|2\pi \sum_h|}}\Big). \tag{8}$$

In the training phase, we adopt the end-to-end training strategy. Given a dataset with N time series, for fixed pipeline configuration and hyperparameters, the neural networks are trained by minimizing an overall loss function containing the aforementioned three functions:

$$\begin{aligned} L_{\text{overall}} = & \frac{1}{N} \sum_{i=1}^{N} D(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_i^{\text{rec}}) + \lambda_1 \frac{1}{N} \sum_{i=1}^{N} L_{\text{self}}(\underline{\boldsymbol{x}}_i) \\ & + \lambda_2 \frac{1}{N} \sum_{i=1}^{N} E(\underline{\boldsymbol{y}}_i), \end{aligned} \tag{9}$$

where λ_1 and λ_2 are two weighting factors governing the trade-off among these three parts and also are optimized in our AutoML framework with range in $\{0.0001, 0.001, 0.01, 0.1, 1\}$. Minimizing reconstruction error $D(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_i^{\text{rec}})$ seeks to construct a one-to-one mapping between the time series and their low-dimensional representations. By minimizing the proposed self-supervised contrastive loss $L_{\text{self}}(\underline{\boldsymbol{x}}_i)$, we seek to acquire the more accurate decision boundary between normal samples and anomalies. And the estimation network can be greatly combined with the representation $\underline{\boldsymbol{y}}_i$ by minimizing the sample energy $E(\underline{\boldsymbol{y}}_i)$, which will promote the generation for mixture-component membership prediction. The aforementioned three functions work together and jointly optimize the neural networks.

In the testing phase, we use the sample energy as the anomaly score. The input samples with high sample energy will be deemed as anomalies.

2) Hyperparameters Optimization: Once the anomaly detection pipeline is constructed, we then emphasize on optimizing the hyperparameters for the given pipeline, which can be expressed as a $\boldsymbol{\Theta}-\mathbf{max}$ task, as given below,

$$\begin{aligned} \{\Theta^{C(t+1)}, \Theta^{D(t+1)}\} = & \max_{\Theta^C, \Theta^D} f(\mathrm{K}^{(t)}, \{\Theta^C, \Theta^D\}) \\ & + \chi_C(\Theta^C) + \chi_D(\Theta^D), \end{aligned}$$
$$\chi_C(\Theta^C) = \begin{cases} 0, & \text{if } \Theta^C \in C \\ -\infty, & \text{else} \end{cases}, \ \chi_D(\Theta^D) = \begin{cases} 0, & \text{if } \Theta^D \in D \\ -\infty, & \text{else} \end{cases}, \tag{10}$$

where C and D denote the feasible region of continuous and discrete hyperparameters, respectively. $f(\cdot)$ is the objective function given in problem (1). $\chi_C(\Theta^C)$ and $\chi_D(\Theta^D)$ are penalty terms that make sure the hyperparameters fall in the feasible region. The above problem can be solved using grid search [46], random search [47], trust-region based derivative-free optimization [48] or Bayesian Optimization (BO) [49]. Unless specified otherwise, we adopt Bayesian Optimization to solve this $\boldsymbol{\Theta}-\mathbf{max}$ task owing to its remarkable effectiveness. The detailed information about Bayesian Optimization we utilized will be presented in Appendix A.

The complete process of the proposed TimeAutoAD is presented in Algorithm 2. It is seen that TimeAutoAD consists of two main stages, i.e., anomaly detection pipeline configuration and hyperparameter optimization. In every iteration of TimeAutoAD, the proposed customized Thompson Sampling is utilized to refine the pipeline configuration at first. After that, Bayesian optimization is invoked to optimize the hyperparameters of the model. Finally, the sampling parameters of the chosen options will be updated according to the performance of the configured pipeline.

In Algorithm 2, $r \sim \text{Bernoulli}(\widetilde{r})$ represents the Bernoulli trial, which can turn the continuous reward $\widetilde{r}$, ranging from 0 to 1 and representing the reward probability, to a binary reward r. There are a few parameters of the proposed AutoML framework, i.e., number of TimeAutoAD iterations L, number of Bayesian Optimization iterations B, pre-defined upper bound f_{upp} and lower bound f_{low} to objective function f, and Beta distribution priors α_0 and β_0. In the experiment, we respectively set $L=100$, $B=30$, $f_{\text{upp}}=1$, $f_{\text{low}}=0.7$, $\alpha_0=10$ and $\beta_0=10$. It is worth mentioning that the proposed TimeAutoAD is robust against to the framework hyperparameter (i.e., f_{upp}, f_{low}, α_0, β_0), which will be discussed in Section IV-C.

B. Self-supervised Contrastive Loss

To further enhance the anomaly detection performance of the configured pipeline, we propose a self-supervised contrastive loss which is integrated into our pipeline as negative sample generation module and auxiliary classification network module. Deldari et al. [50] propose $TS-CP^2$ which is the first Change Point Detection method that employs a contrastive learning strategy though learning an embedded representation which separates pairs of embeddings of time adjacent intervals from pairs of interval embeddings separated across time. Note that the proposed self-supervised contrastive loss is different from contrastive learning [50, 51], which is a learning technique to teach the model to minimize the distance of similar (positive) samples and maximize the distance of

Algorithm 2: Procedure of TimeAutoAD

```
Input: $L$: pre-defined threshold for maximum number of iterations. $B$: pre-defined threshold for maximum Bayesian
    optimization iterations. $\underline{\boldsymbol{\alpha}}_0$ and $\underline{\boldsymbol{\beta}}_0$: pre-defined Beta distribution priors. $f_{\text{upp}}$ and $f_{\text{low}}$: pre-defined upper bound
    and lower bound to objective function $f$.
Set: $\underline{\boldsymbol{\alpha}}_i^{(t)} \in \mathbb{R}^{Q_i}$, $\underline{\boldsymbol{\beta}}_i^{(t)} \in \mathbb{R}^{Q_i}$: the cumulative reward and punishment for $i^{th}$ module for the $t^{th}$ iteration, specifically,
    $\underline{\boldsymbol{\alpha}}_i^{(1)} = \underline{\boldsymbol{\alpha}}_0$ and $\underline{\boldsymbol{\beta}}_i^{(1)} = \underline{\boldsymbol{\beta}}_0$.
for $t = 1, 2, \cdots, L$ do
    for $i = 1, 2, \cdots, M$ do
        Sample $\underline{\boldsymbol{w}}_i \sim \text{Beta}(\underline{\boldsymbol{\alpha}}_i^{(t)}, \underline{\boldsymbol{\beta}}_i^{(t)})$
    end
    Obtain the TimeAutoAD pipeline configuration by solving the following optimization problem:
        $\underset{\text{K}}{\text{maximize}} \sum_{i=1}^{M} (\underline{\boldsymbol{k}}_i)^\top \underline{\boldsymbol{w}}_i \quad \text{subject to } \underline{\boldsymbol{k}}_i \in \{0,1\}^{Q_i}, 1^\top \underline{\boldsymbol{k}}_i = 1, \forall i \in [M],$
    for $b = 1, 2, \cdots, B$ do
        Hyperparameters optimization: Update hyperparameters utilizing Bayesian optimization and the obtained
         objective function value is denoted as $f(\text{K}^{(t)}, \{\Theta_b^{C(t)}, \Theta_b^{D(t)}\})$, where $\text{K}^{(t)}$ denote the module options in $t^{th}$
         iteration and $\{\Theta_b^{C(t)}, \Theta_b^{D(t)}\}$ denote both of the continuous and discrete hyperparameters in $b^{th}$ Bayesian
         optimization iteration in $t^{th}$ iteration.
    end
    Update Beta distribution of the options in the configured pipeline:
    1. Let $f^{(t)} = \max\{f(\text{K}^{(t)}, \{\Theta_b^{C(t)}, \Theta_b^{D(t)}\}), b = 1, \cdots, B\}$ denote the performance of TimeAutoAD model at the
      $t^{th}$ iteration.
    2. Compute the continuous reward $\widetilde{r}$:
        $\widetilde{r} = \max\{0, \frac{f^{(t)} - f_{\text{low}}}{f_{\text{upp}} - f_{\text{low}}}\}$
    3. Obtain the binary reward $r \sim \text{Bernoulli}(\widetilde{r})$.
    for $i = 1, 2, \cdots, M$ do
        $\underline{\boldsymbol{\alpha}}_i^{(t+1)} = \underline{\boldsymbol{\alpha}}_i^{(t)} + \underline{\boldsymbol{k}}_i \cdot r$
        $\underline{\boldsymbol{\beta}}_i^{(t+1)} = \underline{\boldsymbol{\beta}}_i^{(t)} + \underline{\boldsymbol{k}}_i \cdot (1 - r)$
    end
end
Output: A TimeAutoAD model with optimized hyperparameters.
```

different (negative) samples in latent space. While the proposed self-supervised contrastive loss aims to maximize the distance between normal samples and the generated negative samples in latent space. And the proposed contrastive training is somewhat similar to Generative Adversarial Network (GAN) [52, 53] since the proposed negative sample generation and auxiliary classification network can be regarded as the generator and discriminator, respectively. According to [25–27], the representations generated by the encoder have a direct impact on the anomaly detection performance. The anomaly detection performance will be enhanced when the encoder representation learning ability is enhanced. For example, the semi-supervised anomaly detection methods [23, 24] usually have superior performance than unsupervised methods. Such methods utilize a small pool of labeled anomaly samples which will significantly promote the model representation learning capability. Instead, the proposed self-supervised contrastive loss does not require any labeled anomaly samples. We make full use of the internal structure and characteristics of unlabeled data to generate negative samples. In this way, we can acquire more training data, thereby improving the performance of the anomaly detection model.

Given a normal time series $\underline{\boldsymbol{x}}_i \in \mathbb{R}^T$, we can generate the negative sample $\underline{\boldsymbol{x}}_i^{\text{neg}}$ through the proposed three different negative sample generation methods, which are illustrated in Fig. 2 and will be discussed as follows:

a) Negative Sample Generation Method 1: We firstly shift the time series, then randomly choose a segment of original time series and enlarge their values by P times. And P is chosen randomly with the interval $[\min(\underline{\boldsymbol{x}}_i), \max(\underline{\boldsymbol{x}}_i)]$, $\min(\underline{\boldsymbol{x}}_i)$ and $\max(\underline{\boldsymbol{x}}_i)$ represent the minimum and maximum value of input time series $\underline{\boldsymbol{x}}_i$, respectively. The shift size, the size of the chosen segment and P are three hyperparameters which will be automatically optimized in our model.

b) Negative Sample Generation Method 2: We randomly inject noises on a few selected timestamps. The absolute value of the noise term belongs to the interval $[\min(\underline{\boldsymbol{x}}_i), \max(\underline{\boldsymbol{x}}_i)]$. The number of chosen timestamps is a hyperparameter which will be automatically optimized.

c) Negative Sample Generation Method 3: We randomly choose two segments of a time series and exchange their

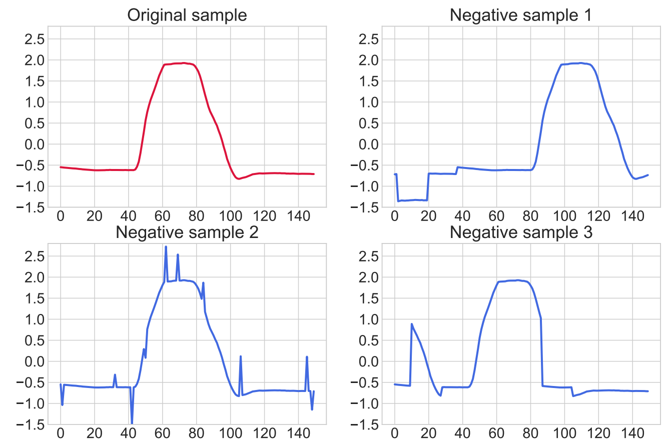

Fig. 2. Illustration of an original sample and three different generated negative samples from GunPoint dataset. Red color curve represents the original sample from GunPoint dataset, and the three Blue curves represent three different negative samples generated according to original sample through three different negative sample generation tricks.

locations. The size of chosen segment is a hyperparameter which will be automatically optimized.

Previous work has shown that the negative sample generation methods of natural language processing can be applied to time series data [54]. For negative sample generation method 1, it is inspired by the generation methods of inconsistent sequences in [55]. The inconsistent sequences can be generated by deleting, permuting, inserting and replacing random tokens in a consistent sequence in [55]. For negative sample generation method 3, it is inspired by the word shuffling method in [56], which shuffles two randomly sampled words in a sentence to generate the fake sentence. As for negative sample generation method 2, it is inspired by the fact that many anomalous samples may contain signal similar to impulse noise in real-word data [57]. Furthermore, it is worth mentioning that all the negative samples generated by the three proposed methods are motivated by examples in real-world data, i.e., they represent real-world anomalies. For the negative samples generated by method 1, for example, they could represent the anomalies in ECG data. Specifically, the shift operation in a normal ECG data could lead to the abnormal PR interval in ECG, which may represent the abnormal impulse transmission from atria to ventricles[1]. The enlarge of a segment in normal ECG data could lead to the abnormal ST segment in ECG data, which may represent the ischemic cardiac disease [58]. For the negative samples generated by method 2, for example, they represent the anomalies contaminated by impulse noise, which can be caused by measurement errors of sensors. For the negative samples generated by method 3, for example, they could represent the anomalies that caused by the clock non-synchronization in wireless sensor networks [59]. Specifically, the anomalies could be the acquired data from the wireless sensor network which have some nodes with imperfect synchronization. In this paper, the proposed TimeAutoAD can automatically select one of the three aforementioned negative sample generation methods, and we generate one negative sample $\underline{\boldsymbol{x}}_i^{\text{neg}}$ for each positive sample $\underline{\boldsymbol{x}}_i$. The proposed contrastive self-supervised loss L_{self} aims to distinguish the positive time series sample from the negative ones, which can be given as:

$$\begin{aligned}
&\underline{\boldsymbol{h}}_i^{\text{pos}} = f_{\text{en}}(\underline{\boldsymbol{x}}_i), \quad \underline{\boldsymbol{h}}_i^{\text{neg}} = f_{\text{en}}(\underline{\boldsymbol{x}}_i^{\text{neg}}),\\
&\underline{\boldsymbol{o}}_i^{\text{pos}} = f_{\text{clas}}(\underline{\boldsymbol{h}}_i^{\text{pos}}), \quad \underline{\boldsymbol{o}}_i^{\text{neg}} = f_{\text{clas}}(\underline{\boldsymbol{h}}_i^{\text{neg}}),\\
&L_{\text{self}}(\underline{\boldsymbol{x}}_i) = f_{\text{BCE}}(\underline{\boldsymbol{o}}_i^{\text{pos}}, l^{\text{pos}}) + f_{\text{BCE}}(\underline{\boldsymbol{o}}_i^{\text{neg}}, l^{\text{neg}}),
\end{aligned} \tag{11}$$

where $\underline{\boldsymbol{h}}_i^{\text{pos}} \in \mathbb{R}^S$ and $\underline{\boldsymbol{h}}_i^{\text{neg}} \in \mathbb{R}^S$ are respectively the latent space representations of positive samples and negative samples, S represents the length of latent space representation. f_{clas} is the auxiliary classification network which is a multi-layer feed-forward neural network with a sigmoid activation function. The input of f_{clas} are $\underline{\boldsymbol{h}}_i^{\text{pos}}$ and $\underline{\boldsymbol{h}}_i^{\text{neg}}$, and the output of f_{clas} are $\underline{\boldsymbol{o}}_i^{\text{pos}} \in \mathbb{R}^1$ and $\underline{\boldsymbol{o}}_i^{\text{neg}} \in \mathbb{R}^1$. f_{BCE} represents binary cross entropy and can be expressed as:

$$\begin{aligned}
&f_{\text{BCE}}(\underline{\boldsymbol{o}}_i^{\text{pos}}, l^{\text{pos}}) = -l^{\text{pos}} \cdot \log(\underline{\boldsymbol{o}}_i^{\text{pos}}) - (1-l^{\text{pos}}) \cdot \log(1-\underline{\boldsymbol{o}}_i^{\text{pos}}),\\
&f_{\text{BCE}}(\underline{\boldsymbol{o}}_i^{\text{neg}}, l^{\text{neg}}) = -l^{\text{neg}} \cdot \log(\underline{\boldsymbol{o}}_i^{\text{neg}}) - (1-l^{\text{neg}}) \cdot \log(1-\underline{\boldsymbol{o}}_i^{\text{neg}}),
\end{aligned} \tag{12}$$

where $l^{\text{pos}} = 0$ and $l^{\text{neg}} = 1$ are the labels for positive time series and negative time series, respectively. Details about the proposed self-supervised loss L_{self} are shown in Fig. 1, we can see that minimizing L_{self} allows the encoder to distinguish the positive samples from the negative samples in the latent space, and consequently enhance the time series anomaly detection ability.

C. Module Options and Hyperparameters in TimeAutoAD

The proposed TimeAutoAD can automatically configure the time series anomaly detection pipeline and optimize its hyperparameters. As mentioned above, the configured anomaly detection pipeline consists of three key parts, i.e., auto representation learning, auto anomaly score calculation and auto negative sample generation. A total of nine modules constitute these three key parts. Each module has its options and the hyperparameters in each option belong to given intervals, which are presented in TABLE I. The detailed explanations of all hyperparameters and their value type are listed in TABLE II. Furthermore, we have conducted ablation study to evaluate the effectiveness of modules in our pipeline, which is presented in Appendix D.

IV. EXPERIMENT

We examine TimeAutoAD for time series anomaly detection from four aspects:

1) **Effectiveness**: Will TimeAutoAD effectively model the temporal dynamic of MTS and capture the unusual patterns?
2) **Robustness**: Can TimeAutoAD maintain its effectiveness in the presence of contaminated training data?
3) **Parameter Sensitivity**: Whether TimeAutoAD is robust against the choice of AutoML framwork parameters?
4) **Visualization**: How can we visualize and interpret the obtained TimeAutoAD model?

[1]https://ecgwaves.com/overview-of-the-ecg-waves-deflections-intervals-durations/

TABLE I
MODULES, OPTIONS, AND HYPERPARAMETERS OF TIMEAUTOAD.

Module	Options	Hyperparameters
Data Augmentation	Scaling	$N^{\text{aug}} \in [0, 100]$, $h^{\text{amp}} \in [0.5, 1.8]$
	Shifting	$N^{\text{aug}} \in [0, 100]$, $h^{\text{shift}} \in [-10, 10]$
	Time-Warping	$N^{\text{aug}} \in [0, 100]$, $h^{\text{tm}} \in [T/10, T/4]$
Negative Sample Generation	Generation Method 1	$P \in [\min(\underline{\boldsymbol{x}}_i), \max(\underline{\boldsymbol{x}}_i)]$, $M^{\text{sh}} \in [T/10, T/4]$, $M^{\text{chos}} \in [T/32, T/8]$
	Generation Method 2	$M^{\text{ts}} \in [T/10, T/4]$
	Generation Method 3	$M^{\text{chos}} \in [T/10, T/4]$
Encoder	RNN / LSTM / GRU	$h^{\text{enc}} \in [1, 32]$
Attention	Self-Attention / No Attention	None
Decoder	RNN / LSTM / GRU	$h^{\text{dec}} \in [1, 32]$ or $[n^{\text{feat}}, 4 * n^{\text{feat}}]$
EM Estimator	Gaussian Mixture Model	$G^{\text{com}} \in [1, 10]$
Similarity Selection	Relative Euclidean Distance / Cosine Similarity / Both	None
Estimation Network	Multi-Layer Feed-Forward Neural Network	$e^{\text{layer}} \in [1, 5]$, $e_i^{\text{node}} \in [8, 128]$
Auxiliary Classification Network	Multi-Layer Feed-Forward Neural Network with Sigmoid Activation Function	$c^{\text{layer}} \in [1, 5]$, $c_i^{\text{node}} \in [8, 128]$

TABLE II
EXPLANATION OF HYPERPARAMETERS IN TABLE I.

Hyperparameters	Meaning	Type
N^{aug}	The number of data augmentation samples	Discrete
h^{amp}	The scaling size of a time series	Continuous
h^{shift}	The shift size of a time series	Discrete
h^{tm}	The number of time-warping timestamps	Discrete
P	Magnification of the chosen segment	Continuous
M^{sh}	The shift size of a time series	Discrete
M^{chos}	The size of the chosen segment	Discrete
M^{ts}	The number of chosen timestamps	Discrete
h^{enc}	The size of encoder hidden state	Discrete
h^{dec}	The size of decoder hidden state	Discrete
n^{feat}	The dimension of multivariate time series	Discrete
G^{com}	The number of mixture-component of GMM	Discrete
e^{layer}	The number of neural network layers in estimation network	Discrete
e_i^{node}	The number of nodes in each layer in estimation network	Discrete
c^{layer}	The number of neural network layers in auxiliary classification network	Discrete
c_i^{node}	The number of nodes in each layer in auxiliary classification network	Discrete
T	The timestamps number of the input time series	Discrete

Datasets and Baselines: TimeAutoAD is utilized for detecting collective anomalies in this paper, which have been studied in many scenarios as ECG time series anomaly detection [25], IoT time series anomaly detection [63], Trace time series anomaly detection [57]. And it is worth mentioning that TimeAutoAD could deal with stream data when choosing a sliding window with a suitable size. We first assess the anomaly detection performance of the proposed TimeAutoAD on several public real-world datasets, i.e., Train-Ticket dataset [57], IoT-23 dataset[2], and HeartBeat dataset[3]. We create the training dataset with only normal samples, and we set the ratio of normal samples to abnormal samples as 10:1 in both validation and testing sets. It is worth mentioning that due to the characteristics of RNN/LSTM/GRU, the proposed TimeAutoAD is able to deal with time series of different lengths.

* Train-Ticket is a train ticket booking system based on a microservice architecture which contains 41 microservices, and the anomalies are added into the 41 microservices to generate anomalous samples [64].
* IoT-23 is a dataset of network traffic from the Internet of Things devices. It has 20 malware captures executed in IoT devices, and 3 captures for benign IoT devices traffic.
* HeartBeat dataset is collected in either a clinical or non-clinical environment, sourced from several contributors around the world. The normal samples represent the recordings that were sourced from healthy contributors and the abnormal samples were from patients with a confirmed cardiac diagnosis.

We next assess the anomaly detection performance of the proposed TimeAutoAD on a multitude of UEA multivariate time series datasets [65] and a total of 85 UCR univariate time series datasets [66] to further assess the performance of the proposed TimeAutoAD. For datasets in UCR and

[2]https://www.stratosphereips.org/datasets-iot23
[3]https://www.physionet.org/physiobank/database/challenge/2016

TABLE III
AUC SCORES OF TIMEAUTOAD AND STATE-OF-THE-ART ANOMALY DETECTION METHODS. BOLD AND UNDERLINED SCORES RESPECTIVELY REPRESENT THE BEST AND SECOND-BEST PERFORMING METHODS.

Model	Train-Ticket	IoT-23	Heartbeat	ECG200	ECGFiveDays	ItalyPD	LSST	RacketSports	PhonemeSpectra
LOF [30]	0.7145	0.6402	0.5527	0.6271	0.5783	0.6061	0.6492	0.4418	0.5646
IF [60]	0.5991	0.6544	0.5329	0.6953	0.6971	0.7510	0.6185	0.5012	0.5355
OCSVM [8]	0.9155	0.5000	0.5799	0.7109	0.6799	0.9016	0.5000	0.5000	0.4444
GRU-ED [61]	0.9148	0.7430	0.6189	0.7001	0.7412	0.8289	$\underline{0.7412}$	0.7163	0.5401
DAGMM [34]	0.8850	0.7964	0.6048	0.5729	0.5732	0.7994	0.5113	0.3953	0.5262
Latent ODE [62]	0.8556	0.7945	0.6577	0.8214	0.6111	0.8221	0.6828	$\underline{0.8232}$	$\underline{0.6813}$
BeatGAN [25]	$\underline{0.9394}$	0.8021	0.6431	$\underline{0.8441}$	$\underline{0.9012}$	$\underline{0.9798}$	0.7296	0.6289	0.4628
DeepSVDD-LSTM [10]	0.9051	$\underline{0.8056}$	$\underline{0.6598}$	0.6585	0.7887	0.7553	0.6285	0.7986	0.6193
TimeAutoAD without L_{self}	0.9669	0.8926	0.7791	0.9442	0.9851	0.9879	0.7804	0.9825	0.8567
TimeAutoAD	**0.9713**	**0.9041**	**0.8031**	**0.9651**	**0.9963**	**0.9959**	**0.7965**	**0.9983**	**0.8817**
Improvement	**3.19%**	**9.85%**	**14.33%**	**12.10%**	**9.51%**	**1.61%**	**5.53%**	**17.51%**	**20.04%**

UEA archives, we follow the strategy in [67] to create the training, validation, and testing sets. Following the previous work [68, 69], we employ AUC (Area Under the Receiver Operating Curve) to evaluate performance, which is a widely-used evaluation metric in anomaly detection. The proposed TimeAutoAD is compared with a set of state-of-the-art methods including Local Outlier Factor (LoF) [30], Isolation Forest (IF) [60], OCSVM [8], GRU-ED [61], DAGMM [34], Latent ODE [62], BeatGAN [25] and DeepSVDD [10]. To allow fair comparison, the neural networks in DeepSVDD are replaced by LSTM, as DeepSVDD-LSTM. Furthermore, we also conduct hyperparameter optimization for each of the state-of-the-art methods.

A. *Effectiveness*

We assess the anomaly detection performance of TimeAutoAD and state-of-the-art anomaly detection methods on three public real-world anomaly detection datasets, i.e., Train-Ticket dataset, IoT-23 dataset, HeartBeat dataset, which are presented in TABLE III. Furthermore, the anomaly detection results of TimeAutoAD on a multitude of multivariate time series are presented in TABLE III. Due to the space limitation, we only present a portion of the results on UCR archive in TABLE III. And the characteristics of each datasets are summarized in Appendix F, TABLE A3. It is seen from TABLE III that the deep learning based methods like GRU-ED and BeatGAN achieve superior performance over traditional anomaly detectors, such as LOF and OCSVM, since such methods do not consider temporal dependencies among time series data. And it is observed that TimeAutoAD outperforms all the competing methods, indicating the superior anomaly detection performance of TimeAutoAD. Moreover, it is worth mentioning that the proposed self-supervised contrastive loss function L_{self} plays an important role, as presented in TABLE III by comparing the TimeAutoAD with and without L_{self}.

In addition, to further assess the superior performance of the proposed TimeAutoAD, we evaluate TimeAutoAD on the remaining univariate time series datasets in UCR archive, which are presented in Appendix B, TABLE A1. It is seen that TimeAutoAD achieves the best anomaly detection performance over the majority $> 90\%$, demonstrating the effectiveness of the proposed model. Please note that solving an AutoML problem incurs additional compuational cost than the traditional model training approach due to the hyper parameter optimization and pipeline configurations, especially for large datasets [70]. The proposed TimeAutoAD is nevertheless computationally efficient for three reasons: 1) TimeAutoAD provides a compact search space containing succinct module options and hyperparameters selections, which has been detailed in Section III-C; 2) TimeAutoAD is composed of highly effective searching procedures, as shown in Algorithm 2, which make the configuration of anomaly detection pipeline and the optimization of hyperparameters less computationally expensive; 3) TimeAutoAD is tailored for time series anomaly detection, the computational complexity of TimeAutoAD is therefore much less than that of the AutoML approach for image data [71, 72]. In fact, in the experiment, the training of TimeAutoAD takes only a few hours on almost all the datasets (e.g., Train-Ticket, IoT-23, ECG200) with a single NVIDIA GeForce GTX TITAN X GPU, asserting its computational efficiency.

TABLE IV
ANOMALY DETECTION RESULTS (AUC SCORES) ON ITALYPOWERDEMAND DATASET WHEN TRAINING DATA ARE CONTAMINATED.

Ratio $c\%$	TimeAutoAD	BeatGAN	GRU-ED
0%	0.9959	0.9798	0.8289
5%	0.9794	0.9374	0.7926
10%	0.9676	0.8827	0.7654
Ratio $c\%$	OCSVM	IF	DeepSVDD
0%	0.9016	0.7510	0.7553
5%	0.8030	0.7252	0.7226
10%	0.7539	0.7011	0.6974

B. Robustness

In this subsection, we investigate how the proposed TimeAutoAD performs in the presence of contaminated training data. We take samples from the normal class mixed with $c\%$ of samples from the anomaly class for model training [34]. TABLE IV reports AUC scores of TimeAutoAD and the baselines when $c = 5\%$ and $c = 10\%$. As we can see, in general, the contaminated training data have a negative effect on anomaly detection performance. We also notice that the proposed TimeAutoAD can maintain excellent detection performance with 10% contaminated training data. To further evaluate the robustness of TimeAutoAD, we carry out experiments on datasets presented in TABLE III, and the results are summarized in TABLE V. It is seen that the anomaly detection performance of TimeAutoAD slightly degrades when training data are contaminated, indicating the robustness of TimeAutoAD.

C. AutoML Framework Parameter Sensitivity Analysis

Although the parameters of the anomaly detection pipeline can be automatically optimized in our AutoML framework, there are a few AutoML prior parameters need to be set.

We analyze the robustness of the proposed method against the choice of Beta distribution priors $\underline{\boldsymbol{\alpha}}_0$ and $\underline{\boldsymbol{\beta}}_0$ in our AutoML framework for each combination of $\underline{\boldsymbol{\alpha}}_0 \in \{5, 10, 15, 20, 25, 30\}$ and $\underline{\boldsymbol{\beta}}_0 \in \{5, 10, 15, 20, 25, 30\}$. We observe that the proposed model is robust against the choice of Beta distribution priors $\underline{\boldsymbol{\alpha}}_0$ and $\underline{\boldsymbol{\beta}}_0$. In the experiment, we set $\underline{\boldsymbol{\alpha}}_0 = 10$ and $\underline{\boldsymbol{\beta}}_0 = 10$, respectively.

We also analyze the robustness of the proposed method against the choice of f_{upp} and f_{low}, we observe that the proposed model is robust against the choice of f_{upp} and f_{low}. And it is necessary to ensure $f_{\text{low}} < f < f_{\text{upp}}$. In the experiment, we set $f_{\text{upp}} = 1$ and $f_{\text{low}} = 0.7$ for most of the datasets. The detailed content about the AutoML framework parameter sensitivity analysis will be discussed in Appendix E.

D. Ablation Study

To verify the efficiency of the proposed AutoML framework, here we present a comparison between TimeAutoAD and TimeAutoAD-. For TimeAutoAD-, both anomaly detection pipeline configuration and hyperparameter search are random. TimeAutoAD and TimeAutoAD- are tested ten times on ECGFiveDays dataset and the result is presented on Fig. 3. It is seen that the performance of the configured pipeline by TimeAutoAD continues to improve and stabilize with the number of iterations grows, while the performance of TimeAutoAD- shows an unstable trend, demonstrating the efficiency of the proposed AutoML framework.

E. Visualization

A synthetic dataset is used to elucidate the underlying mechanism of TimeAutoAD model for detecting time series anomalies. Fig. 4 shows the latent space representation learned via TimeAutoAD model from a synthetic dataset. In this dataset, smooth Sine curves are normal time series. The anomaly time series is created by adding noise to the normal time series over a short interval. It is evident from Fig. 4 that the latent space representations of normal time series lie in a high density area that can be well characterized by a GMM; while the abnormal time series appears to deviate from the majority of the observations in the latent space. In short, the proposed encoder-decoder structure allows us to project the time series data in the original space onto vector representations in the latent space. In this way, we can detect anomalies via clustering-based methods, e.g., GMM, and easily visualize as well as interpret the detected anomalies in time series.

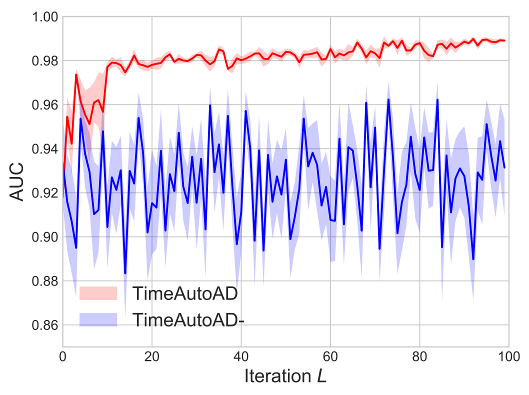

Fig. 3. The illustration of the ablation study of TimeAutoAD.

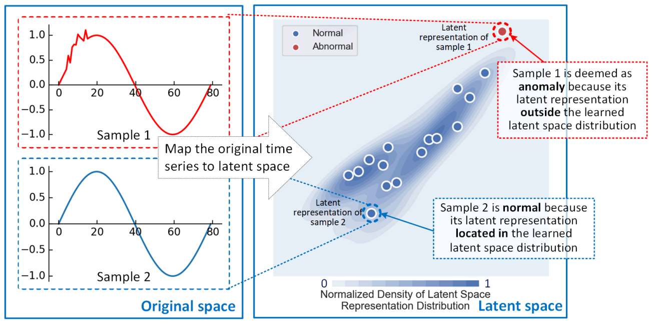
Sample 1
Sample 2
Map the original time series to latent space
Normal
Abnormal
Latent representation of sample 1
Latent representation of sample 2
Sample 1 is deemed as **anomaly** because its latent representation **outside** the learned latent space distribution
Sample 2 is **normal** because its latent representation **located in** the learned latent space distribution
Normalized Density of Latent Space Representation Distribution
Original space
Latent space

Fig. 4. Anomaly interpretation via analysis in latent space. In the left, red and blue curves respectively represent the abnormal and normal sample. In the right, red dot and blue dots respectively represent the latent representation of abnormal sample and normal samples.

V. Conclusion

In this paper, we proposed TimeAutoAD, i.e., an AutoML framework, to carry out unsupervised autonomous anomaly detection for multivariate time series data. Also, we proposed three negative sample generation methods and a novel self-supervised contrastive loss function to enhance anomaly

TABLE V
AUC SCORES OF TIMEAUTOAD WHEN TIME SERIES TRAINING DATASETS ARE CONTAMINATED WITH 5% AND 10% ANOMALY SAMPLES.

Ratio	Train-Ticket	IoT-23	Heartbeat	ECG200	ECGFiveDays	ItalyPD	LSST	RacketSports	PhonemeSpectra
0%	0.9713	0.9041	0.8031	0.9651	0.9963	0.9959	0.7965	0.9983	0.8817
5%	0.9648	0.8921	0.7946	0.9518	0.9842	0.9794	0.7869	0.9878	0.8724
10%	0.9597	0.8853	0.7825	0.9339	0.9759	0.9676	0.7754	0.9752	0.8613
Decline	**1.16%**	**1.88%**	**2.06%**	**3.12%**	**2.04%**	**2.83%**	**2.11%**	**2.31%**	**2.04%**

detection performance. Our empirical studies demonstrated the effectiveness of the proposed TimeAutoAD on a large number of real-world datasets. In the future work, more interesting negative sample generation methods for unsupervised time series anomaly detection could be designed. And semi-supervised learning based autonomous anomaly detection for multivariate time series data could be considered.

APPENDIX A
BAYESIAN OPTIMIZATION

Let $\underline{\boldsymbol{\theta}}$ and $f(\underline{\boldsymbol{\theta}})$ denote the variables needed to be optimized and the objective function, respectively. Given the objective function values during the preceding T iterations $\underline{\boldsymbol{v}} = [f(\underline{\boldsymbol{\theta}}^{(0)}), f(\underline{\boldsymbol{\theta}}^{(1)}), \cdots, f(\underline{\boldsymbol{\theta}}^{(T)})]$, we pick up the variable for sampling in the next iteration via solving the maximization problem that involves the acquisition function i.e., Expected Improvement (EI) based on the posterior Gaussian Process (GP) model.

Specifically, the objective function is assumed to follow a GP model [49] and can be expressed as $f(\underline{\boldsymbol{\theta}}) \sim \mathrm{GP}(m(\underline{\boldsymbol{\theta}}), \boldsymbol{K})$, where $m(\underline{\boldsymbol{\theta}})$ represents the mean function. And $\boldsymbol{K}$ represents the covariance matrix of $\{\underline{\boldsymbol{\theta}}^{(t)}\}_{t=0}^{T}$, namely, $\boldsymbol{K}_{ij} = \kappa(\underline{\boldsymbol{\theta}}^{(i)}, \underline{\boldsymbol{\theta}}^{(j)})$, where $\kappa(\cdot,\cdot)$ is the kernel function. In particular, the poster probability of $f(\underline{\boldsymbol{\theta}})$ at iteration $T+1$ is assumed to follow a Gaussian distribution with mean $\mu(\underline{\boldsymbol{\theta}}^*)$ and covariance $\sigma^2(\underline{\boldsymbol{\theta}}^*)$, given the observation function values $\underline{\boldsymbol{v}}$:

$$\begin{aligned} \mu(\underline{\boldsymbol{\theta}}^*) &= \underline{\boldsymbol{\kappa}}^T[\boldsymbol{K} + \sigma_n^2 \mathrm{I}]^{-1}\underline{\boldsymbol{v}}, \\ \sigma^2(\underline{\boldsymbol{\theta}}^*) &= \kappa(\underline{\boldsymbol{\theta}}^*, \underline{\boldsymbol{\theta}}^*) - \underline{\boldsymbol{\kappa}}^T[\boldsymbol{K} + \sigma_n^2 \mathrm{I}]^{-1}\underline{\boldsymbol{\kappa}}, \end{aligned} \tag{A.1}$$

where $\underline{\boldsymbol{\kappa}}$ is a vector of covariance terms between $\underline{\boldsymbol{\theta}}^*$ and $\{\underline{\boldsymbol{\theta}}^{(t)}\}_{t=0}^{T}$, and σ_n^2 denotes the noise variance. We choose the kernel function as ARD Matérn 5/2 kernel [49] in this paper:

$$\kappa(\underline{\boldsymbol{p}}, \underline{\boldsymbol{p}}') = \tau_0^2 \exp(-\sqrt{5}r)(1 + \sqrt{5}r + \frac{5}{3}r^2), \tag{A.2}$$

where $\underline{\boldsymbol{p}}$ and $\underline{\boldsymbol{p}}'$ are input vectors, $r^2 = \sum_{i=1}^{d} (\underline{\boldsymbol{p}}_i - \underline{\boldsymbol{p}}'_i)^2/\tau_i^2$, and $\psi = \{\{\tau_i\}_{i=0}^{d}, \sigma_n^2\}$ are the GP hyperparameters which are determined by minimizing the negative log marginal likelihood $\log(y|\psi)$:

$$\min_{\psi} \log\det(\boldsymbol{K} + \sigma_n^2 \mathrm{I}) + \underline{\boldsymbol{v}}^T(\boldsymbol{K} + \sigma_n^2 \mathrm{I})^{-1}\underline{\boldsymbol{v}}. \tag{A.3}$$

Given the mean $\mu(\underline{\boldsymbol{\theta}}^*)$ and covariance $\sigma^2(\underline{\boldsymbol{\theta}}^*)$ in (A.1), $\underline{\boldsymbol{\theta}}^{(T+1)}$ can be obtained via solving the following optimization problem:

$$\begin{aligned} \underline{\boldsymbol{\theta}}^{(T+1)} &= \arg\max_{\underline{\boldsymbol{\theta}}^*} \mathrm{EI}(\underline{\boldsymbol{\theta}}^*) \\ &= \arg\max_{\underline{\boldsymbol{\theta}}^*} (\mu(\underline{\boldsymbol{\theta}}^*) - y^+)\Phi(\frac{\mu(\underline{\boldsymbol{\theta}}^*) - y^+}{\sigma(\underline{\boldsymbol{\theta}}^*)}) + \sigma\phi(\frac{\mu(\underline{\boldsymbol{\theta}}^*) - y^+}{\sigma(\underline{\boldsymbol{\theta}}^*)}), \end{aligned} \tag{A.4}$$

where $y^+ = \max[f(\underline{\boldsymbol{\theta}}^{(0)}), f(\underline{\boldsymbol{\theta}}^{(1)}), \cdots, f(\underline{\boldsymbol{\theta}}^{(T)})]$ represents the maximum observation value in the previous T iterations. Φ is normal cumulative distribution function and ϕ is normal probability density function. Through maximizing the EI acquisition function, we seek to improve $f(\underline{\boldsymbol{\theta}}^{(T+1)})$ monotonically after each iteration.

APPENDIX B
RESULTS ON UCR ARCHIVE

In Section IV-A, we have presented a portion of the results on UCR archive in TABLE III. The anomaly detection results for the remaining datasets on UCR archive are summarized in TABLE A1. It is seen that TimeAutoAD achieves best anomaly detection performance over the majority $> 90\%$ of the UCR datasets.

APPENDIX C
DATA AUGMENTATION

In this subsection, we will introduce three data augmentation methods for time series data, which are utilized in this paper.

- **Scaling:** Increasing or decreasing the amplitude of the time series. There are two hyperparametes, the number of data augmentation samples $N^{\mathrm{aug}} \in [0, 100]$ and the scaling size $h^{\mathrm{amp}} \in [0.5, 1.8]$.
- **Shifting:** Cyclically shifting the time series to the left or right. There are two hyperparametes, the number of data augmentation samples $N^{\mathrm{aug}} \in [0, 100]$ and the shift size $h^{\mathrm{shift}} \in [-10, 10]$.
- **Time-warping:** Randomly "slowing down" some timestamps and "speeding up" some timestamps. For each timestamp to "speed up", we delete the data value at that timestamp. For each timestamp to "slow down", we insert a new data value just before that timestamp. There are two hyperparametes, the number of data augmentation samples $N^{\mathrm{aug}} \in [0, 100]$ and the number of time-warping timestamps $h^{\mathrm{tm}} \in [T/10, T/4]$ and T is the number of timestamps in the input data.

TABLE A1
AUC SCORES OF TIMEAUTOAD AND STATE-OF-THE-ART ANOMALY DETECTION METHODS ON UCR TIME SERIES DATASET. BOLD AND UNDERLINED SCORES RESPECTIVELY REPRESENT THE BEST AND SECOND-BEST PERFORMING METHODS.

dataset	**TimeAutoAD**	DeepSVDD-LSTM	Latent ODE	BeatGAN	DAGMM	GRU-ED	IF	LOF	OCSVM
Adiac	**1**	**1**	**1**	**1**	**1**	**1**	0.9375	0.4375	0.8125
ArrowHead	**0.9876**	0.7708	0.8592	0.7923	0.872	0.4008	0.7899	0.442	0.4648
Beef	**1**	**1**	**1**	**1**	**1**	0.8333	**1**	0.4167	0.75
BeetleFly	**1**	**1**	**1**	**1**	**1**	**1**	0.35	0.4	0.5
BirdChicken	**1**	0.9	**1**	0.8	0.9	0.6	0.5	0.4	0.7
Car	**1**	0.8462	**1**	0.6233	0.3346	**1**	0.2854	0.4231	0.5
CBF	**1**	0.8201	0.6573	0.9909	0.7983	0.8606	0.6408	0.9399	0.5
ChlorineConcentration	**0.6653**	0.5649	0.5672	0.5291	0.5724	0.5048	0.5449	0.5899	0.6151
CinCECGTorso	0.8951	0.6511	0.6761	**0.9966**	0.7908	0.4958	0.6749	0.9641	0.5
Coffee	**1**	0.8	**1**	**1**	**1**	0.9333	0.75	0.7167	0.8667
Computers	**0.8354**	0.738	0.744	0.738	0.6563	0.7686	0.468	0.5714	0.6311
CricketX	**1**	0.8423	0.9744	0.8754	0.8123	0.7892	0.7405	0.6282	0.5962
CricketY	**1**	0.9311	0.954	0.9828	0.8997	0.931	0.8161	0.9827	0.5862
CricketZ	**1**	0.6944	0.9583	0.8285	0.6897	0.8333	0.6521	0.6249	0.5
DiatomSizeReduction	**1**	0.9	0.8571	**1**	**1**	0.9913	0.9783	0.9946	0.7989
DistalPhalanxOutlineAgeGroup	**0.9912**	0.7637	0.8333	0.8	0.8333	0.6879	0.7021	0.6858	0.5989
DistalPhalanxOutlineCorrect	**0.8626**	0.6993	0.8333	0.5342	0.6721	0.6193	0.6204	0.7693	0.7427
DistalPhalanxTW	**1**	0.9778	0.9143	**1**	**1**	**1**	0.9643	**1**	0.9524
Earthquakes	**0.8418**	0.8069	0.7421	0.6221	0.5529	0.8033	0.5671	0.5428	0.5
ECG5000	**0.9981**	0.8707	0.5648	0.9923	0.8475	0.8998	0.9304	0.5436	0.7855
ElectricDevices	**0.8427**	0.7206	0.5626	0.8381	0.7172	0.7958	0.5518	0.5528	0.5514
FaceAll	**1**	0.8368	0.7674	0.9821	0.9841	0.9844	0.7639	0.7847	0.5278
FaceFour	**1**	0.8368	**1**	**1**	**1**	0.9286	0.9286	0.4286	0.6786
FacesUCR	**1**	**1**	0.6368	0.9276	0.9065	0.8786	0.6782	0.8296	0.6973
FiftyWords	**0.9971**	0.8275	0.8187	0.9895	0.9901	0.5643	0.9474	0.807	0.7719
Fish	**0.9697**	0.7868	0.9394	0.8523	0.7273	0.5909	0.4772	0.6212	0.5682
FordA	0.6229	0.5229	0.6204	0.5496	0.5619	0.6306	0.4963	0.4708	**0.6393**
FordB	0.6008	0.5055	**0.6212**	0.5999	0.6021	0.5949	0.5949	0.4971	0.5507
GunPoint	**0.9362**	0.8012	0.8479	0.7587	0.4701	0.5657	0.4527	0.5173	0.5659
Ham	**0.8961**	0.6389	0.8579	0.6556	0.7667	0.6358	0.6348	0.6296	0.5
HandOutlines	0.8808	0.6876	0.8362	**0.9031**	0.8524	0.5679	0.7349	0.7413	0.6814
Haptics	**0.8817**	0.5291	0.8579	0.7266	0.6698	0.5826	0.6674	0.5167	0.5
Herring	**1**	0.8333	0.9581	0.8333	0.6528	0.8026	0.7231	0.7105	0.6053
InlineSkate	**0.8556**	0.5987	0.8039	0.65	0.7147	0.5559	0.4223	0.6254	0.5059
InsectWingbeatSound	0.91	0.7212	0.6574	0.9605	**0.9735**	0.7549	0.7861	0.9333	0.6861
LargeKitchenAppliances	**0.8708**	0.5391	0.7703	0.5887	0.5824	0.7975	0.5025	0.5289	0.5135
Lightning2	**1**	0.6591	0.9242	0.6061	0.7574	0.5758	0.909	0.7197	0.5114
Lightning7	**1**	0.8421	**1**	**1**	**1**	**1**	**1**	0.4211	0.7105
Mallat	**0.9996**	0.9212	0.6639	0.9979	0.9701	0.5728	0.8377	0.8811	0.5242
Meat	**1**	0.625	**1**	**1**	0.975	**1**	0.7001	0.7001	0.675
MedicalImages	**0.8021**	0.4379	0.6306	0.6735	0.6473	0.6619	0.6059	0.6035	0.6084
MiddlePhalanxOutlineAgeGroup	**1**	0.9201	0.954	0.9673	0.8512	0.7931	0.7414	0.431	0.6437
MiddlePhalanxOutlineCorrect	**0.8669**	0.5125	0.7355	0.4401	0.7012	0.7013	0.4818	0.5725	0.4979
MiddlePhalanxTW	**1**	0.8505	0.9524	**1**	**1**	**1**	0.9762	**1**	0.9286
MoteStrain	**0.9336**	0.6206	0.7348	0.8201	0.5755	0.7084	0.6217	0.5173	0.5044
NonInvasiveFetalECGThorax1	**1**	0.7639	0.9167	**1**	**1**	**1**	0.9306	0.8611	0.8333
NonInvasiveFetalECGThorax2	**1**	0.8819	0.9028	0.9167	**1**	**1**	0.9722	**1**	0.9028
OliveOil	**1**	**1**	0.9167	0.9167	0.9167	0.9167	0.9583	**1**	0.7917
OSULeaf	**0.9909**	0.4227	0.8864	0.8125	0.8892	0.8352	0.375	0.6823	0.5
PhalangesOutlinesCorrect	**0.7423**	0.5342	0.7049	0.4321	0.5521	0.6625	0.5192	0.6629	0.5532
Phoneme	**0.8849**	0.7978	0.6823	0.7054	0.5826	0.7964	0.4904	0.5943	0.5
Plane	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.4	0.75
ProximalPhalanxOutlineAgeGroup	**0.998**	0.967	0.8024	0.975	0.9723	0.9614	0.82	0.775	0.71
ProximalPhalanxOutlineCorrect	**0.9255**	0.8408	0.6482	0.5823	0.7221	0.9051	0.5348	0.7474	0.6573
ProximalPhalanxTW	**1**	**1**	0.8664	0.9663	0.9623	0.9079	0.8889	0.9311	0.9097
RefrigerationDevices	**0.8629**	0.4596	0.7483	0.7264	0.5722	0.5434	0.4665	0.5714	0.5
ScreenType	**0.8572**	0.7453	0.7453	0.7453	0.5472	0.7686	0.4921	0.5289	0.6815
ShapeletSim	**0.9975**	0.6212	0.9	0.7421	0.5721	0.9728	0.5611	0.5481	0.5
ShapesAll	**1**	**1**	**1**	0.9	0.95	**1**	0.85	0.95	0.65
SmallKitchenAppliances	0.9586	0.8843	0.7151	0.6541	0.7321	**0.9621**	0.6812	0.6563	0.5
SonyAIBORobotSurface1	**0.9998**	0.9246	0.6886	0.9982	0.9834	0.9991	0.8129	0.9731	0.5174
SonyAIBORobotSurface2	**0.9907**	0.7492	0.6211	0.9241	0.8994	0.9236	0.5981	0.7152	0.5111
StarLightCurves	**0.9135**	0.8674	0.5548	0.8083	0.8924	0.8386	0.8161	0.5028	0.5699
Strawberry	0.7805	0.6786	0.6786	0.6786	0.5659	**0.8184**	0.4738	0.4433	0.4328
SwedishLeaf	**0.9512**	0.5682	0.9394	0.6963	0.5758	0.6566	0.6212	0.6212	0.25
Symbols	**0.9987**	0.6869	0.7669	0.9881	0.9762	0.947	0.8025	0.9942	0.6474
SyntheticControl	**1**	0.964	**1**	0.736	0.6524	**1**	0.3299	0.66	0.5
ToeSegmentation1	**0.9437**	0.6381	0.7112	0.8819	0.6264	0.5726	0.5226	0.6708	0.5083
ToeSegmentation2	**0.9907**	0.6809	0.8225	0.9358	0.8243	0.6157	0.5612	0.7021	0.5141
Trace	**1**	**1**	**1**	**1**	**1**	**1**	0.9211	0.4211	0.6316
TwoLeadECG	**0.9959**	0.593	0.6485	0.8759	0.6941	0.8641	0.5967	0.8274	0.6477
TwoPatterns	**0.9996**	0.7229	0.5899	0.9936	0.7163	0.9297	0.5411	0.7371	0.6542
UWaveGestureLibraryAll	**0.9941**	0.8978	0.6487	0.9935	0.9898	0.8106	0.9342	0.7896	0.7217
UWaveGestureLibraryX	**0.7477**	0.6613	0.6136	0.6563	0.6796	0.6009	0.5626	0.4696	0.5852
UWaveGestureLibraryY	**0.9845**	0.9292	0.6256	0.9742	0.9626	0.9357	0.9159	0.6244	0.6293
UWaveGestureLibraryZ	**0.9957**	0.9043	0.6587	0.9897	0.9883	0.9662	0.9161	0.8671	0.6074
Wafer	**0.9903**	0.6763	0.4947	0.9315	0.9586	0.6763	0.9436	0.5599	0.7043
Wine	**1**	0.8395	0.9135	0.8704	0.9074	0.7531	0.4259	0.6689	0.4074
WordSynonyms	**0.9929**	0.8005	0.7862	0.9862	0.9621	0.8245	0.8226	0.8442	0.6857
Worms	**0.9749**	0.653	0.8485	0.8978	0.7677	0.7126	0.5341	0.5896	0.5
WormsTwoClass	**0.8891**	0.4061	0.8312	0.6307	0.6957	0.7591	0.4021	0.4432	0.5795
Yoga	**0.7538**	0.4659	0.5823	0.6883	0.6766	0.5884	0.5421	0.6267	0.547

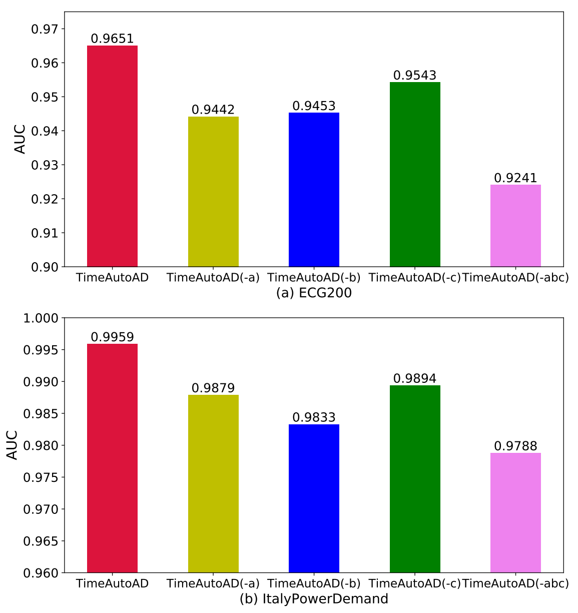

Fig. A1. Ablation study on (a) ECG200 and (b) ItalyPowerDemand dataset. TimeAutoAD(-a) represents TimeAutoAD without negative sample generation and auxiliary classification network modules (i.e. without self-supervised contrastive loss), TimeAutoAD(-b) represents TimeAutoAD without data augmentation module, TimeAutoAD(-c) represents TimeAutoAD without attention module and TimeAutoAD(-a-b-c) represents TimeAutoAD without all the aforementioned modules.

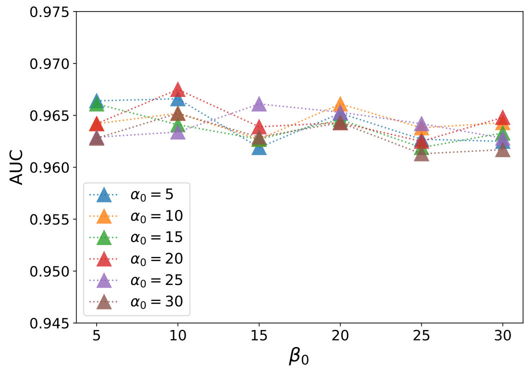

Fig. A2. The sensitivity analysis about Beta distribution priors $\underline{\alpha}_0$ and $\underline{\beta}_0$ in ECG200 dataset, for each combination of $\underline{\alpha}_0 \in \{5, 10, 15, 20, 25, 30\}$ and $\underline{\beta}_0 \in \{5, 10, 15, 20, 25, 30\}$.

Appendix D
Ablation Study

The proposed TimeAutoAD can automatically configure the time series anomaly detection pipeline and optimize its hyperparameters. The anomaly detection pipeline consists of three key parts, i.e., auto representation learning, auto anomaly score calculation and auto negative sample generation. Specifically, encoder, decoder, attention layer, similarity selection, auxiliary classification network and data augmentation modules constitute the auto representation learning module. EM estimator and estimation network modules are used for anomaly score calculation. And the negative sample generation module is employed for negative sample generation. All these modules

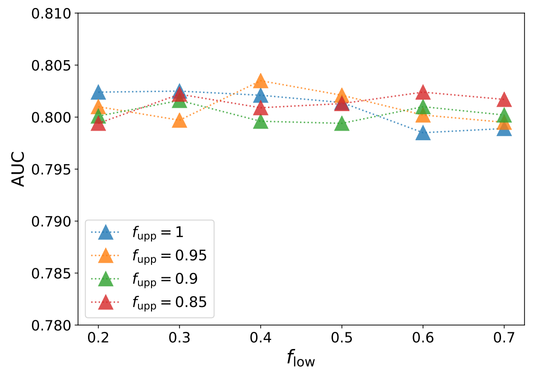

Fig. A3. The sensitivity analysis about the pre-defined upper bound f_{upp} and lower bound f_{low} to objective function f in MedicalImages dataset, for each combination of $f_{\mathrm{upp}} \in \{0.85, 0.9, 0.95, 1\}$ and $f_{\mathrm{low}} \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$.

are integrated together and optimized in an end-to-end manner. In fact, it is revealed through ablation studies that modules in our pipeline can effectively enhance the anomaly detection performance. In particular, we evaluate the effects of data augmentation, attention layer, negative sample generation and auxiliary classification network modules. We conduct experiments on ECG200 and ItalyPowerDemand datasets which are presented in Fig. A1. TimeAutoAD(-a) represents TimeAutoAD without negative sample generation and auxiliary classification network modules (i.e. without self-supervised contrastive loss), TimeAutoAD(-b) represents TimeAutoAD without data augmentation module, TimeAutoAD(-c) represents TimeAutoAD without attention module and TimeAutoAD(-a-b-c) represents TimeAutoAD without all the aforementioned modules. It is seen that the TimeAutoAD(-a-b-c) and TimeAutoAD have the worst and best performance, respectively. And TimeAutoAD(-a), TimeAutoAD(-b) and TimeAutoAD(-c) perform better than TimeAutoAD(-a-b-c), indicating the necessity of these modules in our pipeline.

Furthermore, we evaluate the effects of three negative sample generation methods on different configurations which are presented in TABLE A4. Vanilla represents the anomaly detection pipeline without self-supervised contrastive loss. Method 1/2/3 represent the pipeline with self-supervised contrastive loss and the negative samples are generated by negative sample generation method 1/2/3. We can observe that utilizing self-supervised contrastive loss with negative sample generation method will considerably improve the anomaly detection performance. And it is worth noting that best choice of negative sample generation methods varies for different datasets. This shows that autonomous negative sample generation plays an important role in constructing an effective anomaly detection model.

Appendix E
AutoML Framework Parameter Sensitivity Analysis

We analyze the robustness of the proposed method against the choice of Beta distribution priors $\underline{\alpha}_0$ and $\underline{\beta}_0$ in our

TABLE A2
ABLATION STUDY ABOUT THREE NEGATIVE SAMPLE GENERATION METHODS. THE ANOMALY DETECTION PIPELINE AND HYPERPARAMETERS ARE FIXED. VANILLA REPRESENTS THE ANOMALY DETECTION PIPELINE WITHOUT SELF-SUPERVISED CONTRASTIVE LOSS. METHOD 1/2/3 REPRESENT THE PIPELINE WITH SELF-SUPERVISED CONTRASTIVE LOSS AND THE NEGATIVE SAMPLES ARE GENERATED BY NEGATIVE SAMPLE GENERATION METHOD 1/2/3. BOLD SCORES (AUC SCORES) REPRESENT THE BEST METHODS.

Dataset	Vanilla	Method 1	Method 2	Method 3
TwoLeadECG	0.8472	0.8487	**0.8837**	0.8659
ECGFiveDays	0.7917	0.8081	0.8099	**0.8295**
MoteStrain	0.7968	**0.8175**	0.8168	0.8108
ToeSegmentation2	0.8521	0.8640	**0.8673**	0.8611

AutoML framework, which are tested in ECG200 dataset (average over 10 trials) and the results are shown in Fig. A2. It is seen that, the choice of Beta distribution priors $\underline{\boldsymbol{\alpha}}_0$ and $\underline{\boldsymbol{\beta}}_0$ will not have a significant impact on anomaly detection performance, indicating that the our model is robust against to the choice of Beta distribution priors. Thus, in the experiment, we set $\underline{\boldsymbol{\alpha}}_0 = 10$ and $\underline{\boldsymbol{\beta}}_0 = 10$.

We also analyze the robustness of the proposed method against the choice of f_{upp} and f_{low}, which are tested in MedicalImages dataset (average over 10 trials) and the results are shown in Fig. A3. It is seen that, the choice of f_{upp} and f_{low} will not have a significant impact on model performance. And it is necessary to ensure $f_{\text{low}} < f < f_{\text{upp}}$. In the experiment, we set $f_{\text{upp}} = 1$ and $f_{\text{low}} = 0.7$ for most of the datasets.

TABLE A3
THE CHARACTERISTICS OF DATASETS, INCLUDING THE LENGTH OF MULTIVARIATE TIME SERIES, DIMENSION OF MULTIVARIATE TIME SERIES AND THE NUMBER OF TRAINING, TESTING, VALIDATION INSTANCES.

Datasets	Length	Dimension	Training instances	Testing instances	Validation instances
Train-Ticket	890	1	1000	1100	550
IoT-23	120	1	129	120	79
Heartbeat	405	61	147	96	67
ECG200	96	1	69	71	71
ECGFiveDays	136	1	14	283	188
ItalyPD	24	1	34	339	226
LSST	36	6	777	513	342
RacketSports	51	24	43	48	48
PhonemeSpectra	217	11	85	95	95

APPENDIX F
CHARACTERISTICS OF DATASETS AND CONFIGURED PIPELINE

In this subsection, we will present the characteristics of the datasets in TABLE III. The characteristics include the length of multivariate time series, dimension of multivariate time series (i.e., the number of variables) and the number of training, testing, validation instances, which are summarized in TABLE A3. Furthermore, we present the configured anomaly detection pipelines for some datasets which are shown in TABLE A4.

TABLE A4
THE CONFIGURED ANOMALY DETECTION PIPELINE.

Module	ECGFiveDays	ItalyPD
Data Augmentation	Scaling	Shifting
Negative Sample Generation	Method 2	Method 2
Encoder	GRU	RNN
Attention	Self-Attention	No Attention
Decoder	LSTM	RNN
EM Estimator	GMM	GMM
Similarity Selection	Cosine Similarity	Relative Euclidean Distance
Estimation Network	Multi-Layer Feed-Forward Neural Network	Multi-Layer Feed-Forward Neural Network
Auxiliary Classification Network	Multi-Layer Feed-Forward Neural Network with Sigmoid Activation Function	Multi-Layer Feed-Forward Neural Network with Sigmoid Activation Function

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61771013, in part by the Fundamental Research Funds for the Central Universities of China, and in part by the Fundamental Research Funds of Shanghai Jiading District.

REFERENCES

[1] X. Xu, S. Huang, Y. Chen, K. Browny, I. Halilovicy, and W. Lu, "TSAaaS: Time series analytics as a service on IoT," in *2014 IEEE International Conference on Web Services*, pp. 249–256, IEEE, 2014.

[2] D. W. Dowding, M. Turley, and T. Garrido, "The impact of an electronic health record on nurse sensitive patient outcomes: An interrupted time series analysis," *Journal of the American Medical Informatics Association*, vol. 19, no. 4, pp. 615–620, 2012.

[3] A. C. Rodriguez and M. R. de los Mozos, "Improving network security through traffic log anomaly detection using time series analysis," in *Computational Intelligence in Security for Information Systems 2010*, pp. 125–133, Springer, 2010.

[4] D. Xu, W. Cheng, J. Ni, D. Luo, M. Natsumeda, D. Song, B. Zong, H. Chen, and X. Zhang, "Deep multi-instance contrastive learning with dual attention for anomaly precursor detection," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 91–99, SIAM, 2021.

[5] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1409–1416, 2019.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[7] M. G. Pecht and M. Kang, "Machine learning: Anomaly detection," 2019.

[8] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[9] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[10] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*, pp. 4393–4402, 2018.

[11] C. De Stefano, C. Sansone, and M. Vento, "To reject or not to reject: That is the question-an answer in case of neural classifiers," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 1, pp. 84–94, 2000.

[12] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1253–1258, IEEE, 2019.

[13] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.

[14] X. Jin, E. W. Ma, L. L. Cheng, and M. Pecht, "Health monitoring of cooling fans based on Mahalanobis distance with mRMR feature selection," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 8, pp. 2222–2229, 2012.

[15] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *GI/ITG Workshop MMBnet*, pp. 13–14, 2007.

[16] R. Chitrakar and H. Chuanhe, "Anomaly detection using support vector machine classification with k-medoids clustering," in *2012 Third Asian himalayas international conference on internet*, pp. 1–5, IEEE, 2012.

[17] L. Li, R. J. Hansman, R. Palacios, and R. Welsch, "Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring," *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 45–57, 2016.

[18] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 665–674, 2017.

[19] Y. Chang, Z. Tu, W. Xie, and J. Yuan, "Clustering driven deep autoencoder for video anomaly detection," in *European Conference on Computer Vision*, pp. 329–345, Springer, 2020.

[20] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pp. 385–392, IEEE, 2011.

[21] S. Liu, P. Ram, D. Vijaykeerthy, D. Bouneffouf, G. Bramble, H. Samulowitz, D. Wang, A. Conn, and A. G. Gray, "An ADMM based framework for AutoML pipeline configuration.," in *AAAI*, pp. 4892–4899, 2020.

[22] C. Thornton, F. Hutter, H. H. Hoos, K. Leyton-Brown, *et al.*, "Auto-WEAK: Automated selection and hyper-parameter optimization of classification algorithms," *CoRR, abs/1208.3719*, 2012.

[23] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 353–362, 2019.

[24] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," *arXiv preprint arXiv:1906.02694*, 2019.

[25] B. Zhou, S. Liu, B. Hooi, X. Cheng, and J. Ye, "BeatGAN: Anomalous rhythm detection using adversarially generated time series.," in *IJCAI*, pp. 4433–4439, 2019.

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

[27] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, pp. 5508–5518, 2019.

[28] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13016–13026, 2020.

[29] H. Sun, Y. Bao, F. Zhao, G. Yu, and D. Wang, "CD-trees: An efficient index structure for outlier detection," in *International Conference on Web-Age Information Management*, pp. 600–609, Springer, 2004.

[30] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

[31] Y. Zhong, H. Yamaki, and H. Takakura, "A grid-based clustering for low-overhead anomaly intrusion detection," in *2011 5th International Conference on Network and System Security*, pp. 17–24, IEEE, 2011.

[32] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[33] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

[34] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep Autoencoding Gaussian Mixture Model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018.

[35] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7036–7045, 2019.

[36] H. Xu, L. Yao, W. Zhang, X. Liang, and Z. Li, "Auto-FPN: Automatic network architecture adaptation for object detection beyond classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6649–6658, 2019.

[37] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "NAS-UNET: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44247–44257, 2019.

[38] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pp. 9126–9135, 2019.

[39] Y. Fan, F. Tian, Y. Xia, T. Qin, X.-Y. Li, and T.-Y. Liu, "Searching better architectures for neural machine translation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

[40] J. Chen, K. Chen, X. Chen, X. Qiu, and X. Huang, "Exploring shared structures and hierarchies for multiple NLP tasks," *arXiv preprint arXiv:1808.07658*, 2018.

[41] A. Ukil and S. Bandyopadhyay, "AutoSensing: Automated feature engineering and learning for classification task of time-series sensor signals,"

[42] D. van Kuppevelt, C. Meijer, F. Huber, A. van der Ploeg, S. Georgievska, and V. van Hees, "Mcfly: Automated deep learning on time series," *SoftwareX*, vol. 12, p. 100548, 2020.

[43] V. Olsavszky, M. Dosius, C. Vladescu, and J. Benecke, "Time series analysis and forecasting with automated machine learning on a national ICD-10 database," *International journal of environmental research and public health*, vol. 17, no. 14, p. 4979, 2020.

[44] Y. Li, Z. Chen, D. Zha, K. Zhou, H. Jin, H. Chen, and X. Hu, "AutoOD: Neural architecture search for outlier detection," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 2117–2122, IEEE, 2021.

[45] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *arXiv preprint arXiv:1904.11585*, 2019.

[46] I. Syarif, A. Prugel-Bennett, and G. Wills, "SVM parameter optimization using grid search and genetic algorithm to improve classification performance," *Telkomnika*, vol. 14, no. 4, p. 1502, 2016.

[47] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[48] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. SIAM, 2009.

[49] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[50] S. Deldari, D. V. Smith, H. Xue, and F. D. Salim, "Time series change point detection with self-supervised contrastive predictive coding," in *Proceedings of the Web Conference 2021*, pp. 3124–3135, 2021.

[51] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[52] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[53] X. Feng, D. Song, Y. Chen, Z. Chen, J. Ni, and H. Chen, "Convolutional transformer based dual discriminator generative adversarial networks for video anomaly detection," in *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 5546–5554, 2021.

[54] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning representations for time series clustering," in *Advances in neural information processing systems*, pp. 3781–3791, 2019.

[55] S. Brahma, "Unsupervised learning of sentence representations using sequence consistency," *arXiv preprint arXiv:1808.04217*, 2018.

[56] V. Ranjan, H. Kwon, N. Balasubramanian, and M. Hoai, "Fake sentence detection as a training task for sentence encoding," *arXiv preprint arXiv:1808.03840*, 2018.

[57] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, 2018.

[58] T. Stamkopoulos, K. Diamantaras, N. Maglaveras, and M. Strintzis, "ECG analysis using nonlinear PCA neural networks for ischemia detection," *IEEE Transactions on Signal Processing*, vol. 46, no. 11, pp. 3058–3067, 1998.

[59] N. Wang, H. He, and D. Liu, "A fault-tolerant schema for clock synchronization and data aggregation in WSN," *International Journal of Networked and Distributed Computing*, vol. 1, no. 1, pp. 46–52, 2013.

[60] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008.

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

[62] Y. Rubanova, R. T. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly-sampled time series," in *Advances in Neural Information Processing Systems*, pp. 5320–5330, 2019.

[63] S. Dou, K. Yang, P. Luo, and Y. Jiao, "Unsupervised anomaly detection in heterogeneous network time series with mixed sampling rates," in *Proceedings of IJCAI workshop on AI for Internet of Things*, 2020.

[64] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue, *et al.*, "Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 48–58, IEEE, 2020.

[65] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The UEA mul-

tivariate time series classification archive, 2018," *arXiv preprint arXiv:1811.00075*, 2018.

[66] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR time series classification archive," 2015.

[67] S. E. Benkabou, K. Benabdeslem, and B. Canitia, "Unsupervised outlier detection for time series by entropy and dynamic time warping," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 463–486, 2018.

[68] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6536–6545, 2018.

[69] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2672–2681, 2018.

[70] L. Parmentier, O. Nicol, L. Jourdan, and M.-E. Kessaci, "TPOT-SH: A faster optimization algorithm to solve the AutoML problem on large datasets," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 471–478, IEEE, 2019.

[71] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, pp. 4780–4789, 2019.

[72] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018.

Yang Jiao received the B.S. degree from Central South University, Changsha, China, in 2020. He is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science, Tongji University. His current research interests include machine learning and distributed optimization.

Kai Yang (M'10–SM'19) received the B.Eng. degree from Southeast University, Nanjing, China, the M.S. degree from the National University of Singapore, Singapore, and the Ph.D. degree from Columbia University, New York, NY, USA. He is a Distinguished Professor with Tongji University, Shanghai, China. He was a Technical Staff Member with Bell Laboratories, Murray Hill, NJ, USA, a Senior Data Scientist with Huawei Technologies, Plano, TX, USA, and a Research Associate with NEC Laboratories America, Princeton, NJ, USA. He has also been an Adjunct Faculty Member with Columbia University since 2011. He holds over 20 patents and has been published extensively in leading IEEE journals and conferences. His current research interests include big data analytics, machine learning, wireless communications and signal processing.

Dr. Yang was a recipient of the Eliahu Jury Award from Columbia University, the Bell Laboratories Teamwork Award, the Huawei Technology Breakthrough Award, and the Huawei Future Star Award. The products he has developed have been deployed by Tier-1 operators and served billions of users worldwide. He serves as an Editor for the IEEE INTERNET OF THINGS JOURNAL and the IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. From 2012 to 2014, he was the Vice-Chair of the IEEE ComSoc Multimedia Communications Technical Committee. In 2017, he founded and served as the Chair of the IEEE TCCN Special Interest Group on AI Embedded Cognitive Networks. He has served as the Demo/Poster Co-Chair of the IEEE INFOCOM, the Symposium Co-Chair of the IEEE GLOBECOM, and the Workshop Co-Chair of the IEEE ICME.

Dongjin Song (M'19) is an assistant professor in the Department of Computer Science & Engineering at the University of Connecticut (UConn). His research interests include data science, artificial intelligence, deep learning, time series analysis, graph representation learning, and related applications. Papers describing his research have been published at top-tier data science and artificial intelligence conferences, such as ICML, ICLR, KDD, ICDM, SDM, AAAI, IJCAI, CVPR, ICCV, MM, etc. He won the UConn REP award in 2021, UConn SFF award in 2022, and the best scientific project award in the D4D Challenge in 2013.

Dacheng Tao (F'15) is the Inaugural Director of the JD Explore Academy and a Senior Vice President of JD.com. He mainly applies statistics and mathematics to artificial intelligence and data science, and his research is detailed in one monograph and over 200 publications in prestigious journals and proceedings at leading conferences. He received the 2015 Australian Scopus-Eureka Prize, the 2018 IEEE ICDM Research Contributions Award, and the 2021 IEEE Computer Society McCluskey Technical Achievement Award. He is a fellow of the Australian Academy of Science, AAAS, ACM and IEEE.