A Survey on Time-Series Pre-Trained Models

Qianli Ma, *Member, IEEE,* Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, Zhongzhong Yu, and James T. Kwok, *Fellow, IEEE*

Abstract—Time-Series Mining (TSM) is an important research area since it shows great potential in practical applications. Deep learning models that rely on massive labeled data have been utilized for TSM successfully. However, constructing a large-scale well-labeled dataset is difficult due to data annotation costs. Recently, Pre-Trained Models have gradually attracted attention in the time series domain due to their remarkable performance in computer vision and natural language processing. In this survey, we provide a comprehensive review of Time-Series Pre-Trained Models (TS-PTMs), aiming to guide the understanding, applying, and studying TS-PTMs. Specifically, we first briefly introduce the typical deep learning models employed in TSM. Then, we give an overview of TS-PTMs according to the pre-training techniques. The main categories we explore include supervised, unsupervised, and self-supervised TS-PTMs. Further, extensive experiments are conducted to analyze the advantages and disadvantages of transfer learning strategies, Transformer-based models, and representative TS-PTMs. Finally, we point out some potential directions of TS-PTMs for future work. The source code is available at https://github.com/qianlima-lab/time-series-ptms.

Index Terms—Time-Series Mining, Pre-Trained Models, Deep Learning, Transfer Learning, Transformer

✦

1 INTRODUCTION

As an important research direction in the field of data mining, Time-Series Mining (TSM) has been widely utilized in real-world applications, such as finance [1], speech analysis [2], action recognition [3], [4], and traffic flow forecasting [5], [6]. The fundamental problem of TSM is how to represent the time-series data [7], [8]. Then, various mining tasks can be performed based on the given representations. Traditional time-series representations (e.g., shapelets [9]) are time-consuming due to heavy reliance on domain or expert knowledge. Therefore, it remains challenging to learn the appropriate time series representations automatically.

In recent years, deep learning models [10], [11], [12], [13], [14] have achieved great success in a variety of TSM tasks. Unlike traditional machine learning methods, deep learning models do not require time-consuming feature engineering. Instead, they automatically learn time-series representations through a data-driven approach. However, the success of deep learning models relies on the availability of massive labeled data. In many real-world situations, it can be difficult to construct a large well-labeled dataset due to data acquisition and annotation costs.

To alleviate the reliance of deep learning models on large datasets, approaches based on data augmentation [15], [16] and semi-supervised learning [17] have been commonly used. Data augmentation can effectively enhance the size and quality of the training data, and has been used as an important component in many computer vision tasks [18]. However, different from image data augmentation, time-series data augmentation also needs to consider properties such as temporal dependencies and multi-scale dependen-

cies in the time series. Moreover, design of the time-series data augmentation techniques generally relies on expert knowledge. On the other hand, semi-supervised methods employ a large amount of unlabeled data to improve model performance. However, in many cases, even unlabeled time-series samples can be difficult to collect (e.g., electrocardiogram time series data in healthcare [19], [20]).

Another effective solution to alleviate the problem of insufficient training data is transfer learning [21], [22], which relaxes the assumption that the training and test data must be independently and identically distributed. Transfer learning usually has two stages: pre-training and fine-tuning. During pre-training, the model is pre-trained on some source domains that contain a large amount of data, and are separate but relevant to the target domain. On fine-tuning, the pre-trained model (PTM) is fine-tuned on the often limited data from the target domain.

Recently, PTMs, particularly Transformer-based PTMs, have achieved remarkable performance in various Computer Vision (CV) [23], [24] and Natural Language Processing (NLP) [25] applications. Inspired by these, recent studies consider the design of Time-Series Pre-Trained Models (TS-PTMs) for time-series data. First, a time-series model is pre-trained by *supervised learning* [26], [27], *unsupervised learning* [28], [29], or *self-supervised learning* [30], [31], [32] so as to obtain appropriate representations. The TS-PTM is then fine-tuned on the target domain for improved performance on the downstream TSM tasks (e.g., time-series classification and anomaly detection).

Supervised TS-PTMs [26], [33] are typically pre-trained by classification or forecasting tasks. However, the difficulty to obtain massive labeled time-series datasets for pre-training often limits the performance of supervised TS-PTMs. In addition, unsupervised TS-PTMs utilize unlabeled data for pre-training, which further tackle the limitation of insufficient labeled data. For example, the reconstruction-based TS-PTMs [28] employ auto-encoders and a reconstruction loss to pre-train time-series models. Recently,

Qianli Ma, Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, and Zhongzhong Yu are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: qianlima@scut.edu.cn; cszhenliu@mail.scut.edu.cn; 982360227@qq.com; stevenhuang12@outlook.com; 489531037@qq.com; yuzhong2020@foxmail.com. (Corresponding author: Qianli Ma.)

James T. Kwok is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. E-mail: jamesk@cse.ust.hk.
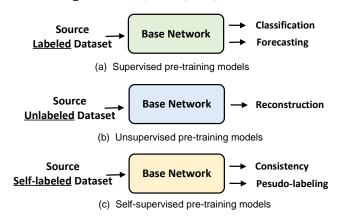
Fig. 1. Pre-training techniques for time series.

self-supervised PTMs based on contrastive learning [34], [35] have shown great potential in CV. Therefore, some scholars [29], [36] have started exploring the design of consistency-based tasks and pseudo-labeling techniques for mining the inherent properties of time series. Nonetheless, the study of TS-PTMs remains a challenge.

In this survey, we provide a comprehensive review of TS-PTMs. Specifically, we first introduce various TSM tasks and deep learning models used in TSM. We then propose a taxonomy of TS-PTMs based on the pre-training techniques (Figure 1). These include supervised pre-training techniques (leading to classification-based and forecasting-based PTMs), unsupervised pre-training techniques (reconstruction-based PTMs) and self-supervised pre-training techniques (consistency-based and pseudo-labeling-based PTMs). Note that some TS-PTMs may use multiple tasks (e.g., forecasting and reconstruction in [37] or consistency in [38]) for pre-training. To simplify the review, we classify the TS-PTM based on its core pre-training task.

Extensive experiments on time-series classification, forecasting and anomaly detection are performed to study the pros and cons of various transfer learning strategies and representative TS-PTMs. In addition, future directions of TS-PTMs are discussed. This survey aims to give readers a comprehensive understanding of TS-PTMs, ranging from the early transfer learning methods to the recent Transformer-based and consistency-based TS-PTMs. The main contributions can be summarized as follows:

- We provide a taxonomy and comprehensive review of existing TS-PTMs based on the pre-training techniques used.
- We perform extensive experiments to analyze the pros and cons of TS-PTMs. For time series classification, we find that transfer learning-based TS-PTMs perform poorly on the UCR time series datasets (containing many small datasets), but achieve excellent performance on other publicly-available large time series datasets. For time series forecasting and anomaly detection, we find that designing a suitable Transformer-based pre-training technique should be the focus of future research on TS-PTMs.
- We analyze the limitations of existing TS-PTMs and suggest potential future directions under (i) datasets,

(ii) transformer, (iii) inherent properties, (iv) adversarial attacks, and (v) noisy labels.

The remainder of this paper is organized as follows. Section 2 provides background on the TS-PTM. A comprehensive review of the TS-PTMs is then given in Section 3. Section 4 presents experiments on the various TS-PTMs. Section 5 suggests some future directions. Finally, we summarize our findings in Section 6.

2 BACKGROUND

In this section, we first describe the TSM tasks in Section 2.1. Section 2.2 then introduces various deep learning models used in TSM. Finally, Section 2.3 discusses why we need to employ the PTMs.

2.1 Time-Series Mining (TSM) Tasks

A time series can be represented as a T-dimensional vector $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, \cdots, \boldsymbol{x}_T]$, where T is the length of the time series, $\boldsymbol{x}_t \in \mathbb{R}^M$ is the value at t-th time step, and M is the number of variables.

2.1.1 Time-Series Classification

In time-series classification [39], a labeled time series dataset is used to train a classifier, which can then be used to classify unseen samples. A labeled time-series dataset with N samples can be denoted as $\mathcal{D} = \{(\boldsymbol{X}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{X}_t, \boldsymbol{y}_t), \ldots, (\boldsymbol{X}_N, \boldsymbol{y}_N)\}$, where $\boldsymbol{X}_t$ can be either a univariate or multivariate time series, and $\boldsymbol{y}_t$ is the corresponding one-hot label vector.

2.1.2 Time-Series Forecasting

Time-Series Forecasting (TSF) [40] aims to analyze the dynamics and correlations among historical temporal data to predict future behavior. TSF models usually need to consider the trend and seasonal variations in the time series ,and also correlations between historical observed values. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T]$ be the historical observations, and H be the desired forecasting horizon, the problem is to predict the future values $[\boldsymbol{x}_{T+1}, \boldsymbol{x}_{T+2}, \ldots, \boldsymbol{x}_{T+H}]$.

2.1.3 Time-Series Clustering

Time-Series Clustering (TSC) [41] aims to partition the time-series dataset $\mathcal{D}$ into a partition K clusters $\{c_1, \ldots, c_K\}$, such that both the similarities among samples from the same cluster and the dissimilarities between samples of different clusters are maximized. While clustering has been successfully used on static data, the clustering of time-series data is more difficult because of the presence of temporal and multi-scale dependencies. TSC helps to discover interesting patterns and enables the extraction of valuable information from complex and massive time series datasets.

2.1.4 Time-Series Anomaly Detection

Time-Series Anomaly Detection (TSAD) [42] aims to identify observations that significantly deviate from the other observations in the time series. It has to learn informative representations from the time series $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T]$, and then derive an anomaly score to determine whether a point $\boldsymbol{x}_t$ or a subsequence $S = [x_p, x_{p+1}, \ldots, x_{p+n-1}]$ (where $n \leq |T|$) is anomalous [43], [44].
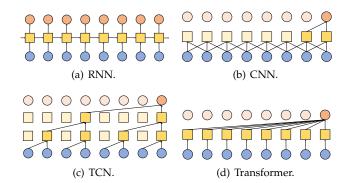
Fig. 2. Deep learning models used for time-series mining.

2.1.5 Time Series Imputation

Time-Series Imputation (TSI) [45] aims to replace missing values in a time series with realistic values so as to facilitate downstream TSM tasks. Given a time series $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T]$ and a binary $\boldsymbol{M} = [m_1, \ldots, m_t, \ldots, m_T]$, $\boldsymbol{x}_t$ is missing if $m_t = 0$, and is observed otherwise. TSI imputes the missing values as:

$$\boldsymbol{X}_{imputed} = \boldsymbol{X} \odot \boldsymbol{M} + \hat{\boldsymbol{X}} \odot (1 - \boldsymbol{M}),$$

where $\hat{\boldsymbol{X}}$ is the predicted values generated by the TSI technique. As a conditional generation model, TSI techniques have been studied [46] and applied to areas such as gene expression [47] and healthcare [48], [49].

2.2 Deep Learning Models for Time Series

2.2.1 Recurrent Neural Networks

A Recurrent Neural Network (RNN) [50] usually consists of an input layer, one or more recurrent hidden layers, and an output layer (Fig. 2(a)). In the past decade, RNNs and their variants (such as the Long Short-Term Memory (LSTM) network [51] and Gated Recurrent Units (GRUs) [50]) have achieved remarkable success in TSM. For example, Muralidhar et al. [52] combined dynamic attention and a RNN-based sequence-to-sequence model for time-series forecasting. Ma et al. [53] employed a multi-layer Dilated RNN to extract multi-scale temporal dependencies for time-series clustering.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [54] are originally designed for computer vision tasks. A typical CNN is shown in Fig. 2(b). To use CNNs for TSM, the data need to be first encoded in an image-like format. The CNN receives embedding of the value at each time step and then aggregates local information from nearby time steps using convolution. CNNs have been shown to be very effective for TSM. For example, the multi-scale convolutional neural network [55] can automatically extract features at different scales by a multi-branch layer and convolutional layers. Kashiparekh et al. [56] incorporated filters of multiple lengths in all convolutional layers to capture multi-scale temporal features for time-series classification.

Unlike vanilla CNNs, Temporal Convolutional Networks (TCNs) [57] use a fully convolutional network [58] so

that all the layers are of the same length, and employ causal convolutions with no information "leakage" from future to past. A typical TCNs is shown in Fig. 2(c). Compared to recurrent networks, TCNs have recently shown to be more accurate, simpler, and more efficient across a diverse range of sequence modeling tasks [59]. For example, Sen et al. [13] combined a local temporal network and a global matrix factorization model regularized by a TCN for time-series forecasting.

2.2.3 Transformers

Transformers [60], [61] integrate information from data points in the time series by dynamically computing the associations between representations with self-attention. A typical Transformer is shown in Fig. 2(d). Transformers have shown great power in TSM due to their powerful capacity to model long-range dependencies. For example, Zhou et al. [14] combined a self-attention mechanism (with $\mathcal{O}(L \log L)$ time and space complexities) and a generative decoder for long time-series forecasting.

2.3 Why Pre-Trained Models?

With the rapid development of deep learning, deep learning-based TSM has received more and more attention. In recent years, deep learning models have been widely used in TSM and have achieved great success. However, as data acquisition and annotation can be expensive, the limited labeled time-series data often hinders sufficient training of deep learning models. For example, in bioinformatics, time series classification involves assigning each sample to a specific category, such as clinical diagnosis results. Nonetheless, obtaining accurate labels for all samples is a challenge as it requires expert knowledge to classify the samples. Therefore, pre-training strategies have been proposed to alleviate this data sparseness problem. The advantages of Pre-Trained Models (PTMs) for TSM can be summarized as follows:

- PTMs provide better model initialization for the downstream TSM tasks, which generally results in better generalization performance.
- PTMs can automatically obtain appropriate time series representations by pre-training on source datasets, thus avoiding over-reliance on expert knowledge.

3 OVERVIEW OF TS-PTMS

In this section, we propose a new taxonomy of TS-PTMs, which systematically classifies existing TS-PTMs based on pre-training techniques. The taxonomy of TS-PTMs is shown in Fig. 3, and please refer to Appendix A.1 for a literature summary of TS-PTMs.

3.1 Supervised PTMs

The early TS-PTMs are inspired by transfer learning applications in CV. Many vision-based PTMs are trained on large labeled datasets such as the ImageNet [62]. The corresponding weights are then fine-tuned on the target dataset, which is usually small. This strategy has been shown to improve
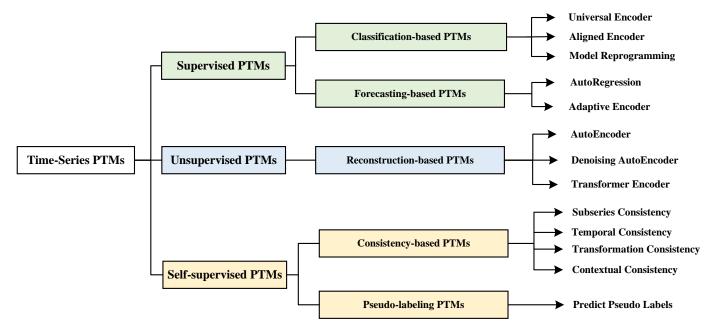
Fig. 3. The taxonomy of Pre-Trained Models for time-series mining.

the generalization performance of deep learning models on many CV tasks. Naturally, some also investigated whether this strategy is effective in the time-series domain [26], [63]. Their experiments on the UCR time series datasets [64] show that transfer learning may improve or degrade downstream task performance, depending on whether the source and target datasets are similar [26].

3.1.1 Classification-based PTMs

Time-series classification is the most common supervised learning task in TSM. The loss function is usually the cross-entropy, which is defined as:

$$\mathcal{L}_{classification} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log(p_{ij}).$$

Here, $\boldsymbol{y}_i = [y_{ij}]$ and $\boldsymbol{p}_i = [p_{ij}]$ are the ground-truth label vector and predicted label vector of the ith input time series, respectively, N is the number of samples and C is the number of categories. Recently, the CNN has also been used for time-series classification [39], [65], [66], and achieved better performance than traditional machine learning methods (such as the nearest neighbor classifier with dynamic time warping distance). However, deep learning models are prone to over-fitting on small-scale time-series datasets.

To prevent the over-fitting problem when training a new model from scratch on the target dataset, there have been attempts to employ classification-based PTMs that pre-train a classification base model on some labeled source datasets [26], [27], [55], [67]. Existing classification-based PTMs can be divided into three categories: (i) universal encoder, (ii) model reprogramming, and (iii) aligned encoder.

Universal Encoder For TS-PTMs, a key issue is how to learn universal time-series representations [68] that can benefit a variety of downstream TSM tasks. A common approach is to design a universal encoder which can be quickly adapted to new tasks with or without fine-tuning. Serrà et
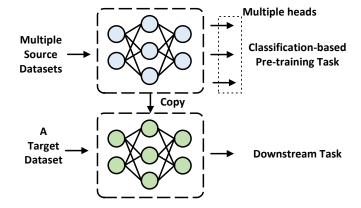

Fig. 4. Universal encoder aims to learn general time-series representations through pre-training on various source datasets. The universal encoder is then fine-tuned on the target dataset for downstream TSM tasks.

al. [63] proposed a universal encoder that combines CNN with attention mechanism, and pre-trained the encoder using the supervised classification task. The encoder is jointly pre-trained on multiple time series datasets. Using multi-head outputs, each dataset has an extra fully-connected layer for classification. In this way, the encoder acts as a carrier to transfer knowledge from multiple related source domains to enhance learning in the target domain. Fig. 4 shows the schematic diagram of the universal encoder.

Fawaz et al. [26] considered transfer learning on time-series data from the shapelet perspective. Shapelets [69], [70] are discriminative subsequences in the time series and can be used as an efficient time-series representation. Fawaz et al. [26] hypothesized that the learned shapelets can generalize to unseen datasets via transfer learning. For each dataset in the UCR archive [64], they trained a fully-convolutional network [65] and then fine-tuned it on all the other datasets. They found that pre-training can degrade (negative transfer)

or improve (positive transfer) the encoder's performance on the target dataset, and the likelihood of positive transfer is greater when the source dataset is similar to the target dataset. Due to privacy and annotation issues, it may be difficult to obtain source datasets that are very similar to the target dataset. To alleviate this problem, Meiseles et al. [71] utilized the clustering property between latent encoding space categories as an indicator to select the best source dataset. The above studies mainly focus on univariate time series. Li et al. [67] proposed a general architecture that can be used for transfer learning on multivariate time series.

The aforementioned works employ CNN as backbone for time series transfer learning. However, vanilla CNNs have difficulty in capturing multi-scale information and long-term dependencies in the time series. Studies [55], [66], [72] have shown that using different time scales or using LSTM with vanilla CNNs can further improve classification performance. For example, Kashiparekh et al. [56] proposed a novel pre-training deep CNN, in which 1-D convolutional filters of multiple lengths are used to capture features at different time scales. Mutegeki et al. [73] used CNN-LSTM as the base network to explore how transfer learning can improve the performance of time-series classification with few labeled time series samples.

Aligned Encoder A universal encoder first pre-trains the model with the source dataset, and then fine-tunes the model using the target dataset. However, the difference between the source and target data distributions is often not considered. To address this issue, some recent works [74], [75] first map the source and target datasets to a shared feature representation space, and then prompt the model to learn domain-invariant representations during pre-training. The pre-training strategy of the aligned encoder is at the core of domain adaptation, and has been extensively validated studied on image data [76]. For time series data, extraction of domain-invariant representations is difficult due to distribution shifts among timestamps and the associative structure among variables. To this end, existing pre-training techniques for time series aligned encoder are based on either Maximum Mean Discrepancy (MMD) [77], [78] or adversarial learning [79], [80].

MMD [81] is a standard metric on distributions, and has been employed to measure the dissimilarity of two distributions in domain adaptation [82]. Given a representation $f(\cdot)$ on source data $\boldsymbol{X}_s \in \mathcal{D}_s$ and target data $\boldsymbol{X}_t \in \mathcal{D}_t$, the empirical approximation of MMD is:

$$\text{MMD}(\mathcal{D}_s, \mathcal{D}_t) = \left\| \frac{1}{|\mathcal{D}_s|} \sum_{\boldsymbol{X}_s \in \mathcal{D}_s} f(\boldsymbol{X}_s) - \frac{1}{|\mathcal{D}_t|} \sum_{\boldsymbol{X}_t \in \mathcal{D}_t} f(\boldsymbol{X}_t) \right\|.$$

MMD-based methods [83], [84] learn domain-invariant representations by minimizing the MMD between the source and target domains in classification training (Fig. 5 (a)). Khan et al. [85] used a CNN to extract features from the source and target domain data separately. The divergence of the source and target domains is reduced by minimizing the Kullback–Leibler divergence in each layer of the network. Wang et al. [83] proposed stratified transfer learning to improve the accuracy for cross-domain activity recognition. Moreover, considering that time lags or offsets
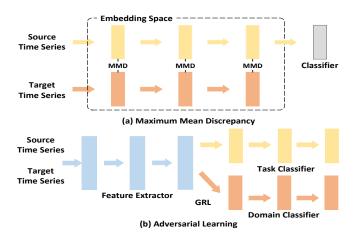

Fig. 5. Aligned encoder aims to learn domain-invariant representations.

can influence the extraction of domain-invariant features, Cai et al. [74] designed a sparse associative structure alignment model which assumes that the causal structures are stable across domains. Li et al. [84] considered the compact causal mechanisms among variables and the variant strength of association, and used the Granger causality alignment model [86] to discover the data's causal structure. Ragab et al. [87] proposed an autoregressive domain discriminator to explicitly address the temporal dependencies during both representation learning and domain alignment. Liu et al. [88] minimized domain divergence on a reinforced MMD metric, which is embedded by a hybrid spectral kernel network in time-series distribution. Despite all these advances, the use of MMD-based methods on time series data is still challenging due to the underlying complex dynamics.

Another common approach is to learn a domain-invariant representation between the source and target domains through adversarial learning [89], [90], [91]. For example, Wilson et al. [75] proposed the Convolutional deep Domain Adaptation model for Time Series data (CoDATS), which consists of a feature extractor, Gradient Reversal Layer (GRL), task classifier, and domain classifier (Fig. 5 (b)). The adversarial step is performed by the GRL placed between the feature extractor and domain classifier in the network. CoDATS first updates the feature extractor and task classifier to classify the labeled source data. The domain classifier is then updated to distinguish which domain each sample comes from. At the same time, the feature extractor is adversarially updated to make it more difficult for the domain classifier to distinguish which domain each sample comes from. Li et al. [92] argued that temporal causal mechanisms should be considered and proposed a time-series causal mechanism transfer network to obtain a domain-invariant representation. However, the exploitation of inherent properties in the time series (such as multi-scale and frequency properties) still need to be further explored in adversarial training.

Model Reprogramming The great success of PTMs in CV [34] and NLP [25] shows that using a large-scale labeled dataset can significantly benefit the downstream tasks. However, most time-series datasets are not large. Recently,
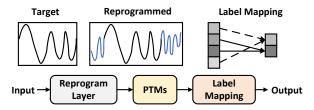
Fig. 6. Model reprogramming can adapt PTMs to a new task by reprogramming the time-series from the target domain and label mapping.

Fig. 7. The RNN-based forecasting architecture.

a novel TS-PTM called Voice2Series [27] was proposed for time-series classification. Voice2Series is based on model reprogramming [93] (Fig. 6). It uses a large acoustic model pre-trained on massive human voice datasets (e.g., spoken-term recognition). The voice data can be considered as univariate time series, and therefore enormous voice data can be employed to pre-train an acoustic model. To make the acoustic model suitable for general time-series data, Voice2Series reprogrammed the model through input transformation learning and output label mapping. The input transformation of a time-series sample $\boldsymbol{X}$ is defined as:

$$\boldsymbol{X}' = \mathrm{Pad}(\boldsymbol{X}) + \boldsymbol{M} \odot \boldsymbol{\theta},$$

where $\mathrm{Pad}(\cdot)$ is a zero-padding function, $\boldsymbol{M}$ is a binary mask, and $\boldsymbol{\theta}$ are reprogramming parameters for aligning the data distributions of the source and target domains. A random (but non-overlapping) many-to-one mapping between source and target labels is used as the output label mapping. Transformer-based attention mechanism [94], [95] is used as the acoustic model. Note that model reprogramming not only uses more labeled training samples during pre-training, but also considers adapts the PTMs to the target tasks by constructing relationships between the source and target domain data.

Summary The universal encoder first pre-trains a base network on the labeled source datasets, and then the base network is transferred to the target domain. This usually requires a large amout of labeled source samples for pre-training, and can be difficult to obtain in the time-series domain. Positive (resp. negative) transfer often occurs when the source and target datasets are similar (resp. dissimilar). Previous studies have explored how to select the source based on inter-dataset similarity or time series representations in the latent representation space. In addition, the aligned encoder based on domain adaption considers the differences between source and target data distributions. Voice2Serie [27] provides a new approach for classification-based PTMs. Some domain-specific time series data (e.g., voice data) is used to pre-train a base network, which is then applied to general time series through model reprogramming. Nevertheless, how to construct a large-scale well-labeled time series dataset suitable for TS-PTMs has not been explored.

3.1.2 Forecasting-based PTMs

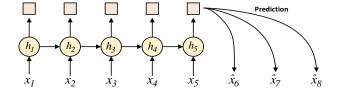Time Series Forecasting (TSF) aims to estimate the values at the future timesteps using observed values from the current and past timesteps. The problem of one-step-ahead forecasting is defined as:

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \boldsymbol{x}_{i,t-k:t}),$$

where $\hat{y}_{i,t+1}$ is the model's predicted value of the ith sample at time $t + 1$, $y_{i,t-k:t} = \{y_{i,t-k}, \ldots, y_{i,t}\}$ and $\boldsymbol{x}_{i,t-k:t} = \{\boldsymbol{x}_{i,t-k}, \ldots, \boldsymbol{x}_{i,t}\}$ are the observed values and exogenous inputs, respectively, over a look-back window of length k, and $f(\cdot)$ is the prediction function learnt by the model [40]. Unlike the classification task that employs manual labels as supervisory information, TSF utilizes the observed value in the future as supervisory information for training. In addition, the mean absolute error or mean squared error is often adopted as the loss function for the TSF task.

A unique property of time-series data is the presence of temporal dependencies. Forecasting can utilize a time series of past and present values to estimate future values, and it is naturally employed as a time series pre-training task to model temporal dependencies. An intuitive approach to produce forecast values is the recursive strategy, which can be achieved by autoregression. In general, a PTM first pre-trains a forecasting-based model on the source dataset. The weights of the base model are then fine-tuned on the target dataset.

AutoRegression Deep learning models, including RNNs [52], TCNs [13] and the more recent Transformers [61], have been used for TSF. The RNN-based forecasting architecture is shown in Fig. 7. In this example, given the historical observations $[\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5]$, TSF tries to predict the future values $[\boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8]$. Thus, TSF does not require manual labels, but uses the values at future timesteps for supervision.

The dynamic nature and inherent temporal dependencies of the time series enable forecasting to assist deep learning models in obtaining robust representations for TSM [37], [96], [97], [98], [99]. For example, Du et al. [97] proposed Adaptive RNNs (AdaRNN) to solve the temporal covariate shift problem that temporal distribution can vary with the statistical properties of the time series. The AdaRNN model consists of Temporal Distribution Characterization (TDC) and Temporal Distribution Matching (TDM) modules. The TDC divides the training time series into least-similar K different subsequences, which fully characterize the distribution information of the time series in each period. The K different subsequences are then used as source data to pre-train a generalized RNN model using the forecasting task. The TDM module can also be used with the Transformer architecture, further improving the TSF performance.

Some recent studies [38], [100], [101] combines the autoregressive forecasting strategy of TSF with contrastive
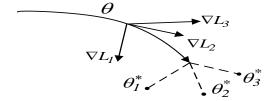
Fig. 8. Adaptive encoder aims to obtain a better initialization model parameters θ so that the model can quickly generalize to new tasks using only a small number of samples (e.g., θ_1^*), where $\theta_i^* (i \in 1, 2, 3)$ denote task adaptive parameters obtained using gradient descent on θ (e.g., ∇L_1) based on the task adaptive data.

Fig. 9. The RNN-based encoder-decoder architecture.

learning [35] to obtain time series representations favorable for downstream tasks. For example, Oord et al. [100] proposed contrastive predictive coding by employing model-predicted timesteps as positive samples and randomly-sampled timesteps as negative samples. Eldele et al. [38] used the autoregressive model's predicted values as positive sample pairs for contrastive learning, thus enabling the model to capture temporal dependencies of the time series. In particular, they fed both strong and weak augmentation samples to the encoder, therefore using a cross-view TSF task as the PTM objective.

Adaptive Encoder Unlike transfer learning which focuses on the current learning ability of the model, meta-learning [102], [103], [104] focuses on the future learning potential of the model to obtain an *adaptive encoder* via task-adaptive pre-training paradigm (as shown in Fig. 8). Especially, transfer learning-based PTMs are prone to overfitting on downstream tasks when the number of samples in the target dataset is small. Inspired by the fact that humans are good at learning a priori knowledge from a few new samples, task-adaptive pre-training, or meta-learning , has also been used. Fig. 8 shows an adaptive encoder via a classic task-adaptive pre-training algorithm called model agnostic meta-learning [105], [106], [107]. Recently, task-adaptive pre-training strategies have been receiving increasing attention in the field of time series. Existing studies focus on how to perform cross-task learning using properties such as long- and short-term trends and multivariate dependencies in the time series.

A task-adaptive pre-training paradigm based on time-series forecasting has been applied to TS-PTMs [108], [109]. For example, Oreshkin et al. [109] proposed a meta-learning framework based on deep stacking of fully-connected networks for zero-shot time-series forecasting. The meta-learning procedure consists of a meta-initialization function, an update function, and a meta-learner. The meta-initialization function defines the initial predictor parameters for a given task. The update function then iteratively updates the predictor parameters based on the previous value of the predictor parameters and the given task. The final predictor parameters are obtained by performing a Bayesian ensemble (or weighted sum) on the whole sequence of parameters. The meta-learner learns shared knowledge across tasks by training on different tasks, thus providing suitable initial predictor parameters for new tasks. Brinkmeyer and Rego [110] developed a few-shot multivariate time series forecasting model working across
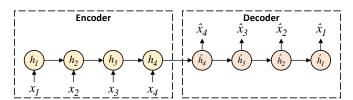
tasks with heterogeneous channels. Experimental results showed that it provides a good generalization for the target datasets. Also, Autoformer [60] and FEDformer [61] show that frequency domain information and Transformer architecture can improve time-series forecasting performance. Hence, it would be an interesting direction to explore task adaptive Transformer-based models for TS-PTMs.

Summary TSF-based PTMs can exploit the complex dynamics in the time series and use that to guide the model in capturing temporal dependencies. Autoregression-based models use the dependencies between subseries and the consistency of future predicted values of the same time series, thus pre-training the time series data using TSF. Unlike classification-based PTMs that use manual labels for pre-training, avoiding sampling bias among subseries (e.g., outliers) [68] for pre-training based on the TSF task remain challenging. Meanwhile, the adaptive encoder based on meta-learning allows for scenarios with small time series samples in the target dataset. In addition, regression-based one-step forecasting models (e.g., RNNs) potentially lead to bad performance due to accumulated error [10], [49]. Instead, some studies [14], [60] employ Transformer-based models to generate all predictions in one forward operation. Therefore, designing efficient TSF encoders would be a basis for studying TSF-based PTMs.

3.2 Unsupervised PTMs

This section introduces unsupervised TS-PTMs, which are often pre-trained by reconstruction techniques. Compared with supervised TS-PTMs, unsupervised TS-PTMs are more widely applicable since they do not require labeled time-series samples.

3.2.1 Reconstruction-based PTMs

Reconstruction is a common unsupervised task, and is usually implemented by an encoder-decoder architecture [53], [111]. The encoder maps the original time series to a latent space of representations, which is then used by the decoder to reconstruct the input time series. The mean square error is often used as the reconstruction loss. For example, Castellani et al. [111] used reconstruction to learn robust representations for detecting noisy labels in the time series. Naturally, reconstruction is also utilized for TS-PTMs. In the following, we introduce TS-PTMs based on the (i) autoencoder, (ii) denoising autoencoder, and (iii) transformer encoder.

AutoEncoder For variable-length NLP sequences such as sentences, paragraphs, and documents, it has been very useful to first convert them to fixed-dimensional vectors
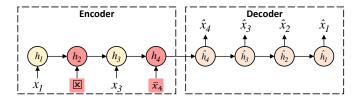
Fig. 10. The RNN-based denoising encoder-decoder architecture.

[25]. Inspired by this, Malhotra et al. proposed the time-series model Timenet [28], which encodes a univariate time series to a fixed-dimensional vector via the Sequence-to-Sequence (Seq2Seq) framework [112]. This consists of an encoder and a decoder. An example RNN-based encoder-decoder is shown in Fig. 9. The encoder converts the input sequence to a fixed-dimensional vector representation, and the decoder reconstructs it to another sequence as output. This process can be written as: $\boldsymbol{H} = f_{enc}(\boldsymbol{X})$ and $\hat{\boldsymbol{X}} = f_{dec}(\boldsymbol{H})$, where $\boldsymbol{H}$ is the representation, $\boldsymbol{X}$ is the input, $\hat{\boldsymbol{X}}$ is the reconstructed input, $f_{enc}(\cdot)$ and $f_{dec}(\cdot)$ are the encoder and decoder functions, respectively. By combining the pre-trained encoder with a classifier, Timenet has shown good performance on time-series classification. However, using autoencoder-based PTMs for other TSM tasks, such as forecasting and anomaly detection, has been less explored.

Denoising AutoEncoder To learn more robust representations for the downstream tasks, the Denoising AutoEncoder (DAE) [113] has been used [114]. As shown in Fig. 10, the input time series $\boldsymbol{X}$ is corrupted to $\tilde{\boldsymbol{X}}$ by adding noise or random masking. The DAE is then trained to reconstruct the original $\boldsymbol{X}$. Compared to the vanilla AutoEncoder, the DAE makes learning more difficult, thus enhancing robustness of the time-series representations. Since the DAE has achieved good performance on representation learning [113], [115], it is also used in the PTMs [32], [116].

A piooneering work that uses DAE in TS-PTM is the Audio Word2Vec [117]. Given variable-length audio segments, Audio Word2Vec obtains fixed-dimensional vectors, which are then used in applications such as query-by-example Spoken Term Detection (STD). To learn robust representations, Audio Word2Vec consists of two stages: offline and online [113]. In the offline phase, a pre-trained model is obtained using all the audio segments. This is then used to encode online language sequences to fixed-dimensional vectors. Inspired by Audio Word2Vec, Hu et al. [118] employed a DAE to pre-train the model for short-term wind speed time-series forecasting. The DAE-based model is pre-trained on massive data from older farms, which is then fine-tuned on data from newly-built farms.

DAE's "mask and predict" mechanism has been successfully used in pre-training NLP [119] and CV models [24], [120], [121]. In the time-series domain, Ma et al. [115] proposed a joint imputation and clustering DAE-based model for incomplete time-series representation learning. They showed that the joint learning strategy enables the imputation task to be optimized in the direction favoring the downstream clustering task, resulting in better downstream clustering performance.

Transformer Encoder   Transformers based on "mask

and predict" training paradigms can be regarded as reconstruction-based models, which have been wildly applied in NLP for studying PTMs [119]. Transformers have received much attention because of their ability to capture input data's contextual (future-past) dependencies, making them a good choice for modeling sequence data [122]. Inspired by Transformer-based PTMs in NLP [123], [124], Zerveas et al. [29] proposed a multivariate time-series unsupervised representation learning model called Time-Series Transformer (TST). Specifically, TST employs the original transformer encoder [125] to learn multivariate time series representations through a designed masking strategy. Further, Shi et al. [116] proposed an unsupervised pre-training method based on the self-attention mechanism of Transformers. In particular, the authors [116] utilize a denoising pretext task that captures the local dependencies and changing trends in the time series by reconstructing corrupted time series. Unlike the above studies, Zhang et al. [32] proposed a cross reconstruction transformer pre-trained by a cross-domain dropping reconstruction task, thereby modeling temporal-spectral relations of time series.

In addition, Transformer-based PTMs have been applied to traffic flow time series, tabular time series, and speech series data. For example, Hou et al. [126] proposed a token-based PTM for traffic flow time series. Concretely, the authors [126] designed a random mask tokens strategy, and then the transformer encoder was pre-trained to predict these tokens. Further, Zhao et al. [127] designed a bidirectional Transformer encoder to learn relationships between two-time intervals across different scales, thus modeling temporal dependencies in traffic flow time series. For tabular time-series data, Padhi et al. [128] proposed hierarchical tabular BERT [119] as the backbone to pre-train by predicting masked tokens. Also, Shankaranarayana et al. [129] utilized a 1-D convolution model combined with the transformer as the backbone, which was pre-trained using the "mask and predict" mechanism. In terms of speech series data, Liu et al. [130] proposed an unsupervised speech representation learning method based on a multilayer transformer encoder, pre-trained by recovering the random 15% zero-masked input frames. Unlike the PTMs mentioned above, Liu et al. [131] proposed a novel PTM called Transformer Encoder Representations from Alteration (TERA) for speech series data. Specifically, TERA utilized three self-supervised training schemes (time alteration, frequency alteration, and magnitude alteration) based on a transformer encoder through reconstructing altered counterpart speech data for pre-training.

Summary DAE-based TS-PTMs add noise or masks to the original time series for pre-training, and have recently been gaining attention compared to autoencoder-based PTMs. Using DAE to study PTMs [24] demonstrates the potential of denoising strategies in pre-training. However, designing DAE-based PTMs applicable to time series is still in the exploratory stage. Meanwhile, Transformer-based PTMs inherit the advantages of unsupervised training from the denoising strategy and have achieved good pre-training results in NLP in recent years [123], [124]. Nevertheless, existing Transformer-based TS-PTMs mainly focus on the time series classification task, and their performance on other

downstream tasks has not been fully explored. At the same time, the time series of different domains may vary widely, resulting in poor model transferability across different domain datasets. Therefore, how to design reconstruction-based unsupervised learning mechanisms (i.e, mask and predict) applicable to different domains of time series is a challenge for studying TS-PTMs.

3.3 Self-supervised PTMs

This section presents self-supervised TS-PTMs based on consistency and pseudo-labeling training strategies which are commonly utilized in self-supervised learning. Compared with unsupervised learning (e.g., reconstruction), self-supervised learning employs self-provided supervisory information (e.g., pseudo-labels) during the training process.

3.3.1 Consistency-based PTMs

Self-supervised learning strategies based on the consistency of different augmented (transformed) views from the same sample have been studied for PTMs in CV [132] with great success. Naturally, consistency-based strategies [100] have recently attracted attention in the time series field. Specifically, consistency-based PTMs keep the distances of different view representations from the same sample (positive pairs) close to each other, while keeping the distances of view representations (negative pairs) from different samples far from each other. The above learning strategy motivates the model to learn representations beneficial for downstream tasks. Based on the above idea, contrastive learning [34], [35] is commonly employed as a training loss for PTMs, which can be defined as:

$$\mathcal{L}_{CL} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(f(\boldsymbol{X}_i)^\mathrm{T} f(\boldsymbol{X}_i^p))}{\exp(f(\boldsymbol{X}_i)^\mathrm{T} f(\boldsymbol{X}_i^p)) + \sum_{j=1}^{B-1} \exp(f(\boldsymbol{X}_i)^\mathrm{T} f(\boldsymbol{X}_i^j))}, \quad (1)$$

where N denotes the number of training sample pairs and each anchor sample ($\boldsymbol{X}_i$) has 1 positive sample ($\boldsymbol{X}_i^p$) and $B - 1$ negative samples. $\boldsymbol{X}_i^j$ denotes the j-th negative sample of the i-th anchor sample.

Consistency-based PTMs need to consider how to construct positive and negative samples so that the model can effectively exploit the complex dynamic properties of time series data. It is worth noting that there is no uniform large-scale well-labeled time series dataset for pre-training. Most existing consistency-based PTMs first pre-train the encoder using the target dataset via self-supervised representation learning, and then fine-tune the pre-trained encoder using the supervised information from the target dataset or directly utilize the representations for downstream tasks. Using contrastive learning for time-series consistency-based PTMs can be roughly divided into four categories [68]: subseries consistency, temporal consistency, transformation consistency, and contextual consistency.

Subseries Consistency   Two subseries belonging to an inclusion relationship sampled from the same time series sample are utilized as positive pair, which is called subseries consistency, as shown in Fig. 11 (a). Utilizing Word2Vec [133] as an analogy, Franceschi et al. first [134] employed a sufficiently long and non-stationary subseries in a time series sample as the context. Then, a subseries selected from the context corresponds to a word (positive subseries), while
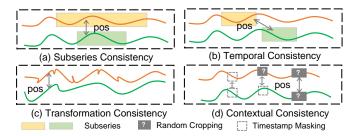

Fig. 11. Positive pair selection strategies in contrastive learning.

a subseries collected from a different context in another time series sample represents a random word (negative subseries). Further, the authors [134] employed the Triplet Loss (T-Loss) to keep the context and positive subsequences close, while making the context and negative subsequences far away for representation learning of time series. Experimental results on UCR and UEA time series archives indicated that the representations obtained by T-Loss can be beneficial for the downstream classification task.

Moreover, Fan et al. [135] proposed a self-supervised time series representation learning framework named Self-Time, by exploring the inter-sample and intra-temporal relations of time series. In terms of the inter-sample relationships, unlike SimCLR [35], the authors employed the cross-entropy loss to guide the reasoning of different levels of entity relationships. As for the intra-temporal relationships, the authors attempted to capture the temporal pattern by reasoning the multi-level relation among subseries sampled from the same time series, thus obtaining representations for the downstream classification task.

Temporal Consistency   Two adjacent subseries from the same time series sample are selected as a positive pair, which is called temporal consistency, as shown in Fig. 11 (b). Based on the assumption of temporal consistency, Tonekaboni et al. [136] proposed a representation learning framework named Temporal Neighborhood Coding (TNC). TNC learns time series representations by ensuring that the distribution of adjacent signals in the coding space is distinguishable from the distribution of non-neighboring signals. Experimental results on multiple time series datasets demonstrate that TNC performs better on downstream classification and clustering tasks compared with the T-Loss [134].

Further, Woo et al. [137] proposed a contrastive learning pre-training method for long sequence time series forecasting, called CoST. Specifically, CoST utilizes temporal consistencies in the time domain to learn the discriminative trend of a sequence, while transforming the data to the frequency domain to discover the seasonal representation of the sequence. Deldari et al. [138] proposed a self-supervised time series change point detection method based on contrast predictive coding. Particularly, the authors exploit local correlations in the time series to drive the maximization of shared information across consecutive time intervals while minimizing shared information across time intervals. Luo et al. [139] applied the objective of contrastive learning to both the local subseries and global instance levels, thus obtaining feature representations favorable to downstream time series forecasting and classification tasks.

Transformation Consistency   Two augmented sequences of the same time series by different transformations are selected as positive pair, which is called transformation consistency, as shown in Fig. 11 (c). Inspired by the successful application of transformation consistency in CV, Eldele et al. [38] proposed a time-series representation learning framework via Temporal and Contextual Contrasting (TS-TCC). TS-TCC first transforms the original time series data in two different ways to obtain weak augmentation (jitter-and-scaling) and strong augmentation (permutation-and-jitter) variants, respectively. Then, the temporal and contextual contrastive learning modules are designed in TS-TCC for learning discriminative representations.

In addition, Hyvärinen et al. [140] analyzed the non-stationarity of time series by performing a nonlinear mixture transformation of subseries from different time windows to ensure that the same subseries are consistent. Unlike [140], Hyvarinen and Morioka [141] developed a method based on a logistic regression estimation model to learn the dependencies of time dimensions by distinguishing the differences between subseries of the original time series and those randomly transformed time points. Lately, Zhang et al. [31] utilized a time-based augmentation bank (jittering, scaling, time-shifts, and neighborhood segments) and frequency-based augmentation methods (adding or removing frequency components) for time series self-supervised contrastive pre-training via time-frequency consistency.

Contextual Consistency   Two augmented contexts with the same timestamp inside the same time series are selected as a positive pair, which is called contextual consistency, as shown in Fig. 11 (d). By designing a timestamp masking and random cropping strategy to obtain augmented contexts, Yue et al. [68] proposed a universal framework for learning time series representations named TS2Vec. Specifically, TS2Vec distinguishes positive and negative samples from the temporal dimensions and instance level hierarchically. In terms of temporal consistency, the augmented contexts with the same timestamp within the same time series are treated as positive pairs, while the augmented contexts in different timestamps are treated as negative pairs. Unlike temporal consistency, instance level consistency treats the augmented contexts of other instances within a batch as negative pairs. Experiments on various time series datasets indicate that TS2Vec achieves excellent performance on downstream classification, forecasting, and anomaly detection tasks.

Recently, Yang et al. [142] proposed a novel representation learning framework for time series, namely Bilinear Temporal-Spectral Fusion (BTSF). Unlike TS2Vec [68], BTSF utilizes instance-level augmentation by simply applying a dropout strategy to obtain positive/negative samples for unsupervised contrastive learning, thus preserving the global context and capturing long-term dependencies of time series. Further, an iterative bilinear temporal-spectral fusion module is designed in BTSF to iteratively refine the representation of time series by encoding the affinities of abundant time-frequency pairs. Extensive experiments on downstream classification, forecasting, and anomaly detection tasks demonstrate the superiority of BTSF.

Summary   The aforementioned studies demonstrate that consistency-based PTMs can obtain robust representations

beneficial for TSM tasks. Although subseries and temporal consistency strategies have achieved good performance on the classification task, sampling biases between subseries (e.g., outliers and pattern shifts) [68] tend to introduce false positive pairs. Meanwhile, the transformation consistency strategy relies on effective instance-level data augmentation techniques. However, designing uniform time series data augmentation techniques for time series datasets from different domains remains a challenge [16], [143]. One alternative is to utilize expert features of time series [144] instead of the commonly used data transformation for contrastive learning. The contextual consistency strategy utilizes a mask (or dropout) mechanism [68], [142] to obtain contrastive pairs by capturing temporal dependencies, which can alleviate the problem of sampling bias among subsequences and achieve excellent performance on forecasting, classification, and anomaly detection tasks. Nevertheless, designing consistency-based strategies using the multi-scale property [66] of time series has not been fully explored.

3.3.2   Pseudo-labeling PTMs

In addition to consistency-based PTMs mentioned above, some other self-supervised TS-PTMs have been explored to improve the performance of TSM. We briefly review these methods in this section.

Predict Pseudo Labels   A large amount of correctly labeled information is the basis for the success of deep neural network models. However, labeling time series data generally requires manual expert knowledge assistance, resulting in high annotation costs. Meanwhile, representation learning using pseudo-label-guided models has yielded rich results in CV [145], [146]. In addition, some studies [147] employed self-supervised learning as an auxiliary task to help the primary task learn better by training the auxiliary task alongside the primary task. In the auxiliary task, when given the transformed sample, the model predicts which transformation is applied to the input (i.e., predicts pseudo labels). This idea has been applied in a few TS-PTMs [135]. For example, Fan et al. [135] randomly selected two length-L subsequences from the same time series and assigned a pseudo-label based on their temporal distance, and then the proposed model is pre-trained by predicting the pseudo-label of subsequence pairs. Furthermore, Zhang et al. [36] incorporated expert features to create pseudo-labels for time series self-supervised contrastive representation learning. Despite these progresses, predicting pseudo labels inevitably contain incorrect labels. How to mitigate the negative impact of incorrect labels will be the focus of studying PTMs.

4   EXPERIMENTAL RESULTS AND ANALYSIS

In this section, following [68], [142], we evaluate TS-PTMs on three TSM tasks, including classification, forecasting, and anomaly detection. Like [68], we select a range of time-series benchmark datasets employed in the corresponding TSM tasks for evaluation. We first analyze the performance of TS-PTMs on the classification task using UCR [148] and UEA [149] archives time series datasets. Also, following [31], we select four time series scenarios datasets for transfer

learning PTMs analysis. Second, the performance of TS-PTMs and related baselines on the forecasting task are compared using the ETT [14] and Electricity [150] datasets. Finally, we analyze the performance of TS-PTMs and related baselines on the anomaly detection task by using the Yahoo [151] and KPI [152] datasets. For information about datasets, baselines, and implementation details, please refer to Appendix A.

PTMs are usually trained in two stages. Firstly, supervised, unsupervised or self-supervised techniques are used to pre-train the base model on the source dataset. Then, the base model is fine-tuned using the training set of the target dataset. Finally, the base model is evaluated on the target test set to obtain the test results. However, there is no available benchmark large-scale well-labelled time series dataset. In other words, selecting a suitable source dataset to pre-train the encoder to obtain a positive transfer performance on the target dataset is difficult. To address the above issues, existing studies (e.g., TS2Vec [68], TST [29], TS-TCC [38], and CoST [137]) utilize self-supervised or unsupervised learning strategies to pre-train the base model using the target dataset and then fine-tune it on the target dataset. We conducted extensive experiments on two fronts to evaluate the effectiveness of existing TS-PTMs. On the one hand, we selected the UCR archive and four scenarios time series datasets for transfer learning using the selected source and target datasets, thus analyzing the performance of different pre-training techniques on the downstream classification task. On the other hand, following the strategy of studies in TS2Vec [68] and CoST [137], we compare and analyze the strategy (using the target dataset to pretrain the base model) on time-series downstream classification, forecasting, and anomaly detection tasks. Through extensive experiments, we aim to provide guidance for the study of pre-training paradigms, techniques for TS-PTMs on different downstream tasks.

4.1 Performance of PTMs on Time-Series Classification

The datasets in the UCR and UEA archives do not specify a validation set for hyperparameter selection, while some datasets have a much higher number of test set samples than the training set. As suggested by [148], we merge the training and test sets for each dataset in UCR and UEA archives. Then, we adopt the five-fold cross-validation strategy to divide the training, validation, and test sets in the ratio of 60%-20%-20%. For the four independent time series scenarios datasets, we utilize the datasets processed by [31] for experimental analysis. Finally, we use the average accuracy on the test set as the evaluation metric.

4.1.1 Comparison of Transfer Learning PTMs based on Supervised Classification and Unsupervised Reconstruction

Advanced time series classification methods (i.e., TS-CHIEF [153], miniRocket [154], and OS-CNN [66]) methods either integrate multiple techniques or require finding the optimal hyperparameters in training and are not suitable as the backbone of PTMs. In recent years, related scholars have employed FCN [26], TCN [68], and Transformer [29] as the backbone for studying TS-PTMs. Considering the training time and classification performance on 128 UCR

time series datasets of the models (please refer to Table 12 in the Appendix), we choose FCN for transfer learning. From the 128 UCR datasets, the 15 datasets with the largest numbers of samples are employed as source datasets. The FCN encoder is pre-trained using the supervised classification task or unsupervised reconstruction task combined with a decoder (symmetric FCN decoder or asymmetric RNN decoder). From the remaining 113 UCR datasets, we select a total of 45 time series datasets as target datasets for downstream classification task fine-tuning. 15 of them have the smallest numbers of samples, 15 of them have the largest numbers of samples, and 15 of them have the medium numbers of samples. Please refer to Tables 9 and 10 in the Appendix for details on the source and target datasets. For the source dataset, we employ all samples to pre-train the FCN encoder. Using the five-fold cross-validation strategy, each target dataset contains five different training sets, and so we performed five fine-tunings for analysis. For each transfer strategy, $15 \times 15 = 225$ sets of transfer results (15 source datasets, 15 target datasets) are obtained for each target dataset sample size (minimum, medium, and maximum). The detailed transfer learning results are shown in Fig. 14 in the Appendix.

Due to space constraints, we report the transfer learning classification results in Table 1. The p-value is calculated for the transfer classification results between the supervised classification and unsupervised reconstruction transfer strategy. As shown in Table 1, the supervised classification transfer (SUP CLS) strategy has the best average Acc and the number of positive transfer results on the minimum, medium and maximum target datasets. The above results indicate that the SUP CLS strategy is more suitable for time series pre-training than the unsupervised transfer strategy. On the smallest target datasets, the overall performance of the SUP CLS strategy and the unsupervised reconstruction strategy utilizing the symmetric FCN decoder is insignificant (P-value greater than 0.05). Also, the unsupervised reconstruction strategy utilizing the symmetric FCN decoder is better than the supervised classification transfer strategy. The above results indicate that a symmetric FCN decoder is more suitable for transfer learning in the time-series classification task than the asymmetric RNN decoder. Overall, the number of positive transfer classification results obtained on the target datasets is unsatisfactory, which may be related to the small number of samples in the UCR source datasets (the number of samples in most source datasets is less than 8000, and please refer to Table 9 in the Appendix).

Further, we select four independent time series datasets for transfer learning PTMs analysis. The number of samples in each source dataset is large, and the test classification results are shown in Table 2. Compared with the supervised approach (FCN) without pre-training, the transfer learning strategy based on supervised classification (Sup CLS) achieves significant positive transfer performance in the neural stage detection and mechanical device diagnosis scenarios. In addition, the transfer learning strategy based on unsupervised RNN Decoders achieves obvious positive transfer performance in the activity recognition scenario. However, in the physical status monitoring scenario, the three transfer learning strategies all result in negative transfer, which may be related to the small number of EMG

samples. Compared with using UCR time series datasets, transfer learning has a better pre-training effect on independent time series datasets with large source datasets.

4.1.2 Comparison of PTMs based on Transfomer and Consistency

We select five TS-PTMs for performance comparison of the downstream classification task. TST [29] is a Transformer-based PTM. T-loss [134], Selftime [135], TS-TCC [38], and TS2Vec [68] are consistency-based PTMs. Specifically, the encoder of the above methods is first pre-trained on the target dataset using an unsupervised/self-supervised pre-learning strategy. Then the label information of the target dataset is utilized to fine-tune the pre-trained encoder or to perform downstream classification tasks using the low-dimensional representations obtained from the pre-trained encoder. Related work in CV [35] and NLP [119] generally utilizes a selected backbone combined with a linear classifier as a supervised method to analyze the classification performance compared with PTMs. Therefore, the FCN combined with a linear classifier is selected as a baseline without pre-training. The averaged results on UCR and UEA archives are reported in Table 3. Detailed results are in Appendix B.2. Since the training of Selftime on the multivariate time series UEA datasets is too time-consuming, we only report its performance on the univariate UCR datasets. Also, we spend about a month on four 1080Ti GPUs to obtain the classification results of SelfTime on the UCR archive.

As shown in Table 3, the classification performance of TS2Vec is the best on the 128 UCR datasets. Also, the P-value is calculated for the classification results between TS2Vec and other methods. The above results show that consistency-based TS2Vec can effectively learn robust representations beneficial for the time-series classification task. However, on the 30 UEA datasets, the model using FCN for direct classification is the best in terms of classification performance and training time, while TS2Vec is inferior in terms of average accuracy and rank. Further, in terms of the P-value for significance tests, TS2Vec is insignificant compared to TS-TCC and TST (P-value greater than 0.05). The UEA archive is the benchmark for multivariate time series classification, while TS2Vec does not have a suitable pre-training strategy designed specifically for time series variables. In addition, the TST model based on the vanilla Transformer has poorer classification results than Supervised (FCN) on both UCR and UEA archives, indicating that the vanilla Transformer architecture is still a challenge for pre-training in the time series classification task.

4.1.3 Visualization

In this section, we use the Class Activation Map (CAM) [155] and heatmap [68] for visualization of the time series classification models. CAM is a way to visualize CNNs and analyze which regions of the input data the CNN-based model focuses on. The heatmap is used in [68] to analyze the heat distribution of the 16-dimensional variables with the largest variance among the feature variables, thus assisting to analyze the trend of the time series. Hence, the ability of the model to capture the change in time series distributions can be measured based on the heat distribution. We employ

Fig. 12. Visualization of the Gunpoint dataset using CAM. For the discriminative features learned by the CNN-based model, red represents high contribution, blue indicates almost no contribution, and yellow is medium contribution (each subplot title gives the accuracy).

the validation sets from the five-fold cross-validation strategy to select models for all comparison settings. Among the five models for analysis, we choose the one with the most significant visualization difference to highlight the visualization.

The Gunpoint dataset in the UCR archive contains two types of series, representing a male or female performing two actions (using a replicate gun or finger to point to the target) [156]. Fawaz et al. [39] employed the Gunpoint dataset for CAM visualization due to its 100% classification accuracy on FCN, low noise, and containing only two types of data. Therefore, we perform a CAM visualization on the Gunpoint dataset, as shown in Fig. 12.

We select the sample with the smallest variance in each class of series in the Gunpoint dataset for the CAM visualization. As shown in Fig. 12, the above two series are discriminative between the two fragments at [45, 60] and [90, 105]. The direct classification using FCN can learn the discriminative subseries information between the two class of series well. However, when the TCN is utilized for direct classification, only one class of series is marked in red (on the fragment [90, 105]), while the other class is marked in blue, resulting in low classification performance. Meanwhile, we select UWaveGestureLibraryX as the source dataset for transfer learning of the Gunpoint dataset. Regarding the supervised classification-based transfer strategy, the FCN model can simultaneously make the two classes of series marked in red for the [90, 105] fragment. For the unsupervised transfer strategy, the FCN model can make one class of series marked in dark red on the [45, 60] and [90, 105] fragments, and another class of series marked in yellow on the [45, 60] and [90, 105] fragments.

The MixedShapesSmallTrain and Wine datasets from the UCR archive are selected as the target datasets to analyze the learning of positive and negative transfer on time-series classification. Based on the classification results of transfer learning in Fig. 14 in the Appendix, we select the source

TABLE 1
Test classification results of transfer learning on 675 sets (15 source datasets multiplied by 45 target datasets) UCR time series datasets. Positive represents the classification performance is better than direct classification without using a transfer strategy.

Transfer Strategy	15 Minimum Target Datasets				15 Medium Target Datasets				15 Maximum Target Datasets			
	Avg Acc	Avg Rank	P-value	Positive	Avg Acc	Avg Rank	P-value	Positive	Avg Acc	Avg Rank	P-value	Positive
Sup CLS	**0.7720 (0.0738)**	1.75	-	**48 (225)**	**0.7196 (0.0398)**	**1.45**	-	**9 (225)**	**0.7885 (0.0231)**	**1.73**	-	**17 (225)**
Unsup FCN Decoder	0.7616 (0.0756)	**1.59**	5.53E-02	41 (225)	0.7001 (0.0380)	1.92	7.64E-08	4 (225)	0.7715 (0.0240)	2.02	1.27E-03	15 (225)
Unsup RNN Decoder	0.7292 (0.0823)	2.16	2.28E-9	29 (225)	0.6655 (0.0355)	2.20	1.87E-15	2 (225)	0.7532 (0.0264)	2.20	9.38E-07	14 (225)

TABLE 2
Test classification results of transfer learning on four independent time series scenarios. Supervised (FCN) means that the FCN encoder is directly used for supervised classification training on the target dataset.

Scenario	Source Dataset	Target Dataset	Supervised (FCN)	Sup CLS	Unsup FCN Decoder	Unsup RNN Decoder
Neurological Stage Detection	SleepEEG	Epilepsy	0.7766 (0.0552)	**0.8021 (0.0010)**	0.6813 (0.2483)	0.6813 (0.2486)
Mechanical Device Diagnosis	FD-A	FD-B	0.5781 (0.0292)	**0.6718 (0.0783)**	0.6541 (0.0547)	0.6541 (0.0543)
Activity Recognition	HAR	Gesture	0.3904 (0.0281)	0.3796 (0.0257)	0.3517 (0.0363)	**0.4933 (0.0196)**
Physical Status Monitoring	ECG	EMG	**0.9756 (0.0322)**	0.4634 (0.0010)	0.8439 (0.0892)	0.8731 (0.0103)

TABLE 3
Comparisons of classification test accuracy using Transformer and consistency-based methods (standard deviations are in parentheses).

Models	128 UCR datasets				30 UEA datasets			
	Avg Acc	Avg Rank	P-value	Training Time (hours)	Avg Acc	Avg Rank	P-value	Training Time (hours)
Supervised (FCN)	0.8296 (0.0520)	2.73	2.59E-03	4.87	**0.7718 (0.0722)**	**1.73**	5.61E-03	**0.73**
T-Loss	0.8325 (0.0302)	3.28	6.60E-04	37.56	0.5863 (0.0459)	4.33	1.52E-04	22.03
Selftime	0.8017 (0.0339)	4.38	3.49E-15	-	-	-	-	-
TS-TCC	0.7807 (0.0430)	4.30	2.76E-11	9.45	0.7059 (0.0432)	3.10	4.09E-01	34.35
TST	0.7755 (0.0365)	4.15	3.46E-11	196.72	0.6921 (0.0382)	3.23	2.91E-01	52.10
TS2Vec	**0.8691 (0.0265)**	**2.07**	-	**0.87**	0.7101 (0.0411)	2.57	-	2.07

(a) MixedShapesSmallTrain   (b) Wine

Fig. 13. Comparison of positive and negative transfer based on supervised classification using the MixedShapesSmallTrain and Wine dataset.

datasets with the best accuracy for positive transfer and the worst accuracy for negative transfer. From Fig. 13, the heat distribution of the variable with the largest 16-dimensional variance obtained by direct classification and positive transfer is more similar to the trend of the original time series, while the negative transfer is difficult to present a heat distribution that matches the trend of the original time series. The heatmap visualization indicates that negative transfer may make it difficult for the model to capture the dynamic change information of the original time series, leading to degraded classification performance.

4.2 Performance of PTMs on Time-Series Forecasting

We further evaluate the performance of PTMs including TS2Vec [68] and CoST [137] on time-series forecasting. Experiments on state-of-the-art direct forecasting methods are also conducted for comparison. These approaches include three Transformer based models, LogTrans [157],

Informer [14], Autoformer [60], and Temporal Convolutional Network (TCN) [57]. TS2Vec and CoST performs pre-training the downstream forecasting task on the same dataset, while other baselines are trained directly on the dataset without pre-training. We employ the mean square error (MSE) and mean absolute error (MAE) as evaluation metrics following [137]. Also, we follow [68] to preprocess the datasets. Table 4 presents the test results on four public datasets for multivariate time-series forecasting. As can be seen, TS2Vec and CoST generally outperform the TCN, LogTrans, and Informer. Autoformer achieves remarkable improvements and has better long-term robustness. Note that CoST achieves comparable performance to Autoformer for some settings on the ETTm1 and Electricity datasets. We credit it to the modeling of trend and seasonal features, which is also verified to be effective in the decomposition architecture of Autoformer. Most existing studies are end-to-end supervised methods, while few utilize pre-training and downstream fine-tuning paradigms for time-series forecasting. The empirical results show that it is a promising paradigm. Although recent time series forecasting techniques such as FiLM [11], Scaleformer [158], and MICN [159] have demonstrated impressive performance, it is important to note that these methods incorporate various time series characteristics (such as frequency domain and multiscale properties) which differ from the single property-based approaches of TS2Vec and CoST. Therefore, we do not adopt the aforementioned methods as baselines. Moreover, mining the seasonal and trend characteristics of time series [61], [160], and utilizing Transformer-based TS-PTMs to discover the temporal dependencies are the potential to improve forecasting performance.

TABLE 4
Comparisons of forecasting test results on ETT and Electricity datasets using TCN, Transformer-based and pre-training methods.

Models		LogTrans		TCN		Informer		Autoformer		TS2Vec		CoST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	0.6149	0.5823	0.6476	0.5765	0.5972	0.5680	**0.3729**	**0.4176**	0.5952	0.5313	0.3861	0.4289
	48	0.8360	0.7075	0.7417	0.6292	0.7344	0.6460	**0.4296**	**0.4419**	0.6316	0.5566	0.4358	0.4634
	168	1.0119	0.8084	0.9388	0.7324	0.9414	0.7480	**0.4796**	**0.4687**	0.7669	0.6405	0.6415	0.5817
	336	1.0821	0.8458	1.1151	0.8305	1.1217	0.8375	**0.4939**	**0.4815**	0.9419	0.7334	0.8099	0.6776
	720	1.0381	0.8167	0.9641	0.7802	1.1921	0.8591	**0.5112**	**0.5100**	1.0948	0.8098	0.9678	0.7700
ETTh2	24	0.7676	0.6854	0.9376	0.7760	1.2772	0.9142	**0.2933**	**0.3689**	0.4478	0.5032	0.4463	0.5032
	48	1.2485	0.9004	1.2930	0.9183	1.4506	0.9663	**0.3349**	**0.3922**	0.6460	0.6184	0.7022	0.6397
	168	5.4984	1.8806	4.0537	1.6939	5.6799	1.9599	**0.4627**	**0.4638**	1.7771	1.0569	1.5428	0.9822
	336	4.4323	1.6926	5.3142	2.0093	4.6730	1.8000	**0.4753**	**0.4794**	2.1157	1.1759	1.7560	1.0478
	720	3.1966	1.4905	9.1991	2.2230	3.6218	1.5993	**0.4970**	**0.5015**	2.5823	1.3521	1.9716	1.0938
ETTm1	24	0.1576	0.2780	0.2003	0.3345	0.2123	0.3418	0.1473	0.2610	0.1970	0.3179	**0.1331**	**0.2589**
	48	0.2533	0.3658	0.2432	0.3641	0.3220	0.4262	**0.1897**	**0.2898**	0.2682	0.3784	0.1959	0.3192
	96	0.4699	0.5022	0.3856	0.4599	0.4427	0.5227	0.3424	**0.3546**	0.3735	0.4496	**0.3043**	0.4034
	288	1.2230	0.8335	0.8695	0.7408	1.1282	0.8074	**0.3264**	**0.3663**	0.7566	0.6672	0.7554	0.6609
	672	3.0955	1.3050	1.9628	1.1394	3.0900	1.3206	**0.4497**	**0.4357**	1.8217	1.0452	1.5897	0.9840
Electricity	24	0.3667	0.4054	0.4616	0.4673	0.5750	0.5159	0.1888	0.3031	0.3038	0.3836	**0.1438**	**0.2471**
	48	0.3842	0.4182	0.4581	0.4694	0.5808	0.5281	0.2043	0.3148	0.3311	0.4014	**0.1602**	**0.2611**
	168	0.3911	0.4206	0.5079	0.4938	0.5486	0.5220	0.2446	0.3401	0.3581	0.4195	**0.1776**	**0.2739**
	336	0.4065	0.4300	0.5209	0.5024	0.6021	0.5571	0.2551	0.3464	0.3865	0.4367	**0.2134**	**0.3018**
	720	0.4118	0.4308	0.5505	0.5116	0.7690	0.6517	0.3075	0.3808	0.4313	0.4618	**0.2629**	**0.3384**

4.3 Performance of PTMs on Time-Series Anomaly Detection

For time-series anomaly detection, we follow the settings of [68], [152] to determine whether the last time point x_t of a sequence fragment $[x_1, x_2, \ldots, x_t]$ is an anomaly. Also, we choose F1-Score, Precision, and Recall as evaluation metrics, and preprocess the datasets in the manner of [68]. The TS2Vec [68] and the benchmark methods SPOT [161], DSPOT [161], LSTM-VAE [162], DONUT [163], Spectral Residual (SR) [152], and Anomaly Transformer (AT) [164] are employed as comparison methods. Like the classification and forecasting tasks, TS2Vec performs pre-training and the downstream anomaly detection task on the same dataset, while other baselines are trained without pre-training. The test results on Yahoo and KPI datasets are shown in Table 5. Among them, the authors of SR do not provide open-source code. We follow the anomaly detection results provided in the original SR article [152] for comparative analysis. As shown in Table 5, TS2Vec outperforms other comparison methods on F1-Score on both the Yahoo and KPI datasets, indicating that the pre-training strategy based on contrastive learning can effectively improve the performance of the time-series anomaly detection task. However, on the Yahoo dataset, AT significantly outperforms TS2Vec in the Precision metric, and the F1-Score is also very close to TS2Vec, indicating that the AT model based on Transformer architecture has certain advantages in the time-series anomaly detection. In addition, the performance of AT demonstrates that Transformer has excellent potential for studying time-series anomaly detection.

5 FUTURE DIRECTIONS

5.1 Large-scale Time Series Datasets

Large-scale benchmark datasets are critical to the success of PTMs in CV and NLP, yet there are no publicly avail-

TABLE 5
Comparisions test results of time-series anomaly detection.

Models	Yahoo			KPI		
	F1-score	Precision	Recall	F1-score	Precision	Recall
SPOT	0.5037	0.4972	0.5105	0.1911	0.7628	0.1092
DSPOT	0.3952	0.3222	0.5109	0.1250	0.0889	0.2106
LSTM-VAE	0.5386	0.4764	0.6196	0.1833	0.3715	0.1216
DONUT	0.4352	0.3795	0.5101	0.4044	0.5423	0.3225
SR*	0.5630	0.4510	0.7470	0.6220	0.6470	**0.5980**
AT	0.7529	**0.8173**	0.6980	0.4444	0.7272	0.3200
TS2Vec	**0.7574**	0.7543	**0.7605**	0.6904	**0.9221**	0.5517

able large-scale datasets in the field of time series. Most existing TS-PTMs are pre-trained on time-series datasets from archives such as UCR [148] and UEA [149], most of which have small sample sizes (only a few thousand or even hundreds of samples). Although these time-series benchmark datasets have contributed significantly to the development of the time-series community, it is challenging to utilize them to pre-train deep learning models due to limitations of sample sizes and generality.

Recently, Yang et al. [27] proposed a TS-PTM called Voice2Series that pre-trains deep learning models using a large-scale sequence dataset from the speech domain. Then, the pre-training models are transferred to the general time-series datasets (UCR time-series archive) through model reprogramming and label mapping. Experimental results indicate that pre-training with large-scale sequence datasets in the speech domain can achieve state-of-the-art on the time-series classification task. Therefore, how to employ large-scale sequence data in related fields is a worthy research direction for TS-PTMs. In addition, the construction of large-scale generic time-series datasets like ImageNet [62] is a crucial focus, which will significantly facilitate further research on TS-PTMs.

5.2 Inherent Properties of Time Series

Time-series representation learning has attracted much attention in recent years. Existing studies have explored the inherent properties of time series for representation learning, such as using CNNs to capture multi-scale dependencies, RNNs to model temporal dependencies, and Transformers to model long-term temporal dependencies. Also, the context-dependencies and frequency domain (or seasonal-trend [137]) information of time series have been explored in recent contrastive learning studies. Although mining the inherent properties of time series can learn representations beneficial for downstream TSM tasks, the transferability of time series from different domains remains weakly interpretable.

Due to the inherent properties (i.e., frequency domain information) of time series, the pre-training techniques applied to image data are difficult to transfer directly to time series. Compared to NLP, TS-PTMs are challenging to learn universal time-series representations due to the absence of a large-scale unified semantic sequence dataset. For example, each word in a text sequence dataset has similar semantics in different sentences with high probability. Therefore, the word embeddings learned by the model can transfer knowledge across different text sequence data scenarios. However, time series datasets are challenging to obtain subsequences (corresponding to words in text sequences) that have consistent semantics across scenarios, making it difficult to transfer the knowledge learned by the model. Hence, exploiting the inherent properties of time series to mine transferable segments in time series data is a challenge for future research on TS-PTMs.

5.3 Transformer in Time Series

Transformer-based models have achieved excellent performance in various fields, such as NLP and CV. So naturally, transformers have also been explored in the study of time series. Transformer employs a multi-head attention mechanism to capture long-term dependencies in the input data. With this advantage, Informer [14], Autoformer [60], and FEDformer [61] for time-series forecasting, and Anomaly Transformer [164] for time-series anomaly detection are well suited for analyzing time series. However, there are relatively few existing Transformer models with competitive advantages for the time-series classification task, which may be the fact that the time-series classification task is more focused on capturing discriminative subseries (e.g., shapelets) or multiscale features of time-series, where existing CNN-based models [66] may be more advantageous.

Although there has been some work applying transformers to TSM tasks [165], there is little research on PTMs for time series. We believe that the current challenges with pre-training Transformer models for time series are two-fold. First, pre-training transformers usually require massive data for pre-training to learn the universal representations. However, there is currently a lack of large-scale datasets in the time series field. Second, how to design an effective pre-trained transformer model combined with the inherent properties of time series still needs further exploration. Therefore, exploring pre-training transformer models for time series is an exciting research direction.

5.4 Adversarial Attacks on Time Series

Adversarial example attacks have recently received extensive attention in various fields because they have significant security risks. Naturally, scholars in the time-series domain have also begun to consider the impact of adversarial sample attacks on time-series models [166], [167], [168], [169]. For example, Karim et al. [170] utilized a distilled model as a surrogate that simulated the behavior of the attacked time-series classification model. Then, an adversarial transformation network is applied to the distilled model to generate time-series adversarial examples. Experiments on 42 UCR datasets show they are vulnerable to adversarial examples.

Adversarial examples are generated by adding perturbations to the original examples to cross the classification boundaries of the classifier. In general, adding random perturbations is difficult to generate adversarial examples. Also, adversarial examples are not easy to generate when each cluster is far from the classification boundary. Recently, Hendrycks et al. [171] found that self-supervised learning could effectively improve the robustness of deep learning models to adversarial examples. Therefore, improving the robustness of time series models to adversarial examples by utilizing TS-PTMs is a direction worth exploring.

5.5 Pre-Training Models for Time-Series Noisy Labels

The acquisition cost of large-scale labeled datasets is very expensive. Therefore, various low-cost surrogate strategies are provided to collect labels automatically. For example, many weakly labeled images can be collected with the help of search engines and crawling algorithms. In the time-series domain, Castellani et al. [111] employed sensor readings to generate labels for time series. Although these strategies enable obtaining large-scale labeled data, they also inevitably lead to label noise. Training deep learning models efficiently with noisy labels is challenging since deep learning models have a high capacity for fitting noisy labels [172]. To this end, many studies on learning with label noise [173] have emerged. As an effective representation learning method, PTMs can be effectively employed to solve the problem of noisy labels [174], [175], which has been studied in CV. However, only a few studies are currently investigating the time-series noisy label [111], and PTMs for time-series noisy labels have not been studied.

6 CONCLUSION

In this survey, we provide a systematic review and analysis of the development of TS-PTMs. In the early research on TS-PTMs, related studies were mainly based on CNN and RNN models for transfer learning on PTMs. In recent years, Transformer-based and consistency-based models have achieved remarkable performance in time-series downstream tasks, and have been utilized for time series pre-training. Hence, we conducted a large-scale experimental analysis of existing TS-PTMs, transfer learning strategies, Transformer-based time series methods, and related representative methods on the three main tasks of time-series classification, forecasting, and anomaly detection. The experimental results indicate that Transformer-based PTMs have significant potential for time-series forecasting and anomaly detection tasks, while designing suitable

Transformer-based models for the time-series classification task remains challenging. Meanwhile, the pre-training strategy based on contrastive learning is a potential focus for the development of future TS-PTMs.

ACKNOWLEDGMENTS

REFERENCES

[1] Y. Wu, J. M. Hernández-Lobato, and G. Zoubin, "Dynamic covariance models for multivariate financial time series," in *International Conference on Machine Learning*. PMLR, 2013, pp. 558–566.

[2] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2020, pp. 6074–6078.

[3] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Twenty-fourth International Joint Conference on Artificial Intelligence*, 2015.

[4] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2891–2900.

[5] S. Wang, J. Cao, and P. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE transactions on knowledge and data engineering*, pp. 1–20, 2020.

[6] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, and A. K. Qin, "A survey on modern deep neural network for traffic prediction: Trends, methods and challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 04, pp. 1544–1561, 2022.

[7] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[8] E. Eldele, M. Ragab, Z. Chen, M. Wu, C.-K. Kwoh, and X. Li, "Label-efficient time series representation learning: A review," *arXiv preprint arXiv:2302.06433*, 2023.

[9] G. Li, B. K. K. Choi, J. Xu, S. S. Bhowmick, K.-P. Chun, and G. L. Wong, "Efficient shapelet discovery for time series classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 03, pp. 1149–1163, 2022.

[10] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.

[11] T. Zhou, Z. Ma, Q. Wen, L. Sun, T. Yao, R. Jin *et al.*, "Film: Frequency improved legendre memory model for long-term time series forecasting," *arXiv preprint arXiv:2205.08897*, 2022.

[12] M. H. Tahan, M. Ghasemzadeh, and S. Asadi, "Development of fully convolutional neural networks based on discretization in time series classification," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–12, 2022.

[13] R. Sen, H.-F. Yu, and I. S. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," in *Advances in Neural Information Processing Systems*, 2019.

[14] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of Advancement of Artificial Intelligence Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[15] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021, pp. 4653–4660.

[16] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *Plos one*, vol. 16, no. 7, p. e0254841, 2021.

[17] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[19] Q. Ma, Z. Zheng, J. Zheng, S. Li, W. Zhuang, and G. W. Cottrell, "Joint-label learning by dual augmentation for time series classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8847–8855.

[20] L. Yang, S. Hong, and L. Zhang, "Spectral propagation graph network for few-shot time series classification," *arXiv preprint arXiv:2202.04769*, 2022.

[21] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[22] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[24] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[25] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, pp. 1–26, 2020.

[26] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1367–1376.

[27] C.-H. H. Yang, Y.-Y. Tsai, and P.-Y. Chen, "Voice2series: Reprogramming acoustic models for time series classification," *International Conference on Machine Learning*, 2021.

[28] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pre-trained deep recurrent neural network for time series classification," *arXiv preprint arXiv:1706.08838*, 2017.

[29] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2114–2124.

[30] S. Deldari, H. Xue, A. Saeed, J. He, D. V. Smith, and F. D. Salim, "Beyond just vision: A review on self-supervised representation learning on multimodal and temporal data," *arXiv preprint arXiv:2206.02353*, 2022.

[31] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik, "Self-supervised contrastive pre-training for time series via time-frequency consistency," in *Advances in neural information processing systems*, 2022.

[32] W. Zhang, L. Yang, S. Geng, and S. Hong, "Cross reconstruction transformer for self-supervised time series representation learning," *arXiv preprint arXiv:2205.09928*, 2022.

[33] R. Ye and Q. Dai, "A novel transfer learning framework for time series forecasting," *Knowledge-Based Systems*, vol. 156, pp. 74–99, 2018.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

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

[36] H. Zhang, J. Wang, Q. Xiao, J. Deng, and Y. Lin, "Sleeppriorcl/: Contrastive representation learning with prior knowledge-based positive mining and adaptive temperature for sleep staging," *arXiv preprint arXiv:2110.09966*, 2021.

[37] N. Laptev, J. Yu, and R. Rajagopal, "Reconstruction and regression loss for time-series transfer learning," in *Proceedings of the Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) and the 4th Workshop on the Mining and LEarning from Time Series (MiLeTS)*, 2018.

[38] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, "Time-series representation learning via temporal and contextual contrasting," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021, pp. 2352–2359.

[39] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[40] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[41] B. Lafabregue, J. Weber, P. Gançarski, and G. Forestier, "End-to-end deep representation learning for time series clustering: A comparative study," *Data Mining and Knowledge Discovery*, pp. 1–53, 2021.

[42] F. Liu, X. Zhou, J. Cao, Z. Wang, T. Wang, H. Wang, and Y. Zhang, "Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional lstm-cnn," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 06, pp. 2626–2640, 2022.

[43] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014.

[44] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021.

[45] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," *arXiv preprint arXiv:1805.10572*, 2018.

[46] Y. Luo, X. Cai, Y. Zhang, J. Xu, and X. Yuan, "Multivariate time series imputation with generative adversarial networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 1603–1614.

[47] M. C. De Souto, I. G. Costa, D. S. De Araujo, T. B. Ludermir, and A. Schliep, "Clustering cancer gene expression data: A comparative study," *BMC Bioinformatics*, vol. 9, no. 1, pp. 1–14, 2008.

[48] J. de Jong, M. A. Emon, P. Wu, R. Karki, M. Sood, P. Godard, A. Ahmad, H. Vrooman, M. Hofmann-Apitius, and H. Fröhlich, "Deep learning for clustering of multivariate clinical patient trajectories with missing values," *GigaScience*, vol. 8, no. 11, p. giz134, 2019.

[49] Q. Ma, S. Li, and G. Cottrell, "Adversarial joint-learning recurrent neural network for incomplete time series classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 4, pp. 1765–1776, 2022.

[50] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[51] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[52] N. Muralidhar, S. Muthiah, and N. Ramakrishnan, "Dyat nets: Dynamic attention networks for state forecasting in cyber-physical systems." in *International Joint Conference on Artificial Intelligence*, 2019, pp. 3180–3186.

[53] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning representations for time series clustering," *Advances in Neural Information Processing Systems*, vol. 32, pp. 3781–3791, 2019.

[54] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[55] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[56] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "Convtimenet: A pre-trained deep convolutional neural network for time series classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[57] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[58] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[59] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *Neurocomputing*, vol. 399, pp. 491–501, 2020.

[60] J. Xu, J. Wang, M. Long *et al.*, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[61] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," *arXiv preprint arXiv:2201.12740*, 2022.

[62] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[63] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series." in *Artificial Intelligence Research and Development*, 2018, pp. 120–129.

[64] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[65] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

[66] W. Tang, G. Long, L. Liu, T. Zhou, M. Blumenstein, and J. Jiang, "Omni-scale cnns: a simple and effective kernel size configuration for time series classification," in *International Conference on Learning Representations*, 2022.

[67] F. Li, K. Shirahama, M. A. Nisar, X. Huang, and M. Grzegorzek, "Deep transfer learning for time series data based on sensor modality classification," *Sensors*, vol. 20, no. 15, p. 4271, 2020.

[68] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," *Proceedings of Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2022.

[69] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 947–956.

[70] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 392–401.

[71] A. Meiseles and L. Rokach, "Source model selection for deep learning in the time series domain," *IEEE Access*, vol. 8, pp. 6190–6200, 2020.

[72] F. J. O. Morales and D. Roggen, "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations," in *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, 2016, pp. 92–99.

[73] R. Mutegeki and D. S. Han, "Feature-representation transfer learning for human activity recognition," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2019, pp. 18–20.

[74] R. Cai, J. Chen, Z. Li, W. Chen, K. Zhang, J. Ye, Z. Li, X. Yang, and Z. Zhang, "Time series domain adaptation via sparse associative structure alignment," in *Proceedings of the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2021.

[75] G. Wilson, J. R. Doppa, and D. J. Cook, "Multi-source deep domain adaptation with weak supervision for time-series sensor data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1768–1778.

[76] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[77] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[78] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua, "Homm: Higher-order moment matching for unsupervised domain adaptation," in *Proceedings of the Advancement of Artificial Intelligence conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 3422–3429.

[79] S. Xie, Z. Zheng, L. Chen, and C. Chen, "Learning semantic representations for unsupervised domain adaptation," in *International conference on machine learning*. PMLR, 2018, pp. 5423–5432.

[80] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-

adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[81] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo, "Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2272–2281.

[82] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Universal domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2720–2729.

[83] J. Wang, Y. Chen, L. Hu, X. Peng, and S. Y. Philip, "Stratified transfer learning for cross-domain activity recognition," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2018, pp. 1–10.

[84] Z. Li, R. Cai, T. Z. Fu, and K. Zhang, "Transferable time-series forecasting under causal conditional shift," *arXiv preprint arXiv:2111.03422*, 2021.

[85] M. A. A. H. Khan, N. Roy, and A. Misra, "Scaling human activity recognition via deep learning-based domain adaptation," in *2018 IEEE international conference on pervasive computing and communications (PerCom)*. IEEE, 2018, pp. 1–9.

[86] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. B. Fox, "Neural granger causality," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4267–4279, 2021.

[87] M. Ragab, E. Eldele, Z. Chen, M. Wu, C.-K. Kwoh, and X. Li, "Self-supervised autoregressive domain adaptation for time series data," *arXiv preprint arXiv:2111.14834*, 2021.

[88] Q. Liu and H. Xue, "Adversarial spectral kernel matching for unsupervised time series domain adaptation," in *International Joint Conference on Artificial Intelligence*, 2021, pp. 2744–2750.

[89] S. Purushotham, W. Carvalho, T. Nilanon, and Y. Liu, "Variational recurrent adversarial deep domain adaptation," in *International Conference on Learning Representations*, 2017.

[90] P. R. d. O. da Costa, A. Akçay, Y. Zhang, and U. Kaymak, "Remaining useful lifetime prediction via deep domain adaptation," *Reliability Engineering & System Safety*, vol. 195, p. 106682, 2020.

[91] W. Garrett, R. D. Janardhan, and J. C. Diane, "Calda: Improving multi-source time series domain adaptation with contrastive adversarial learning," *arXiv preprint arXiv:2109.14778*, 2021.

[92] Z. Li, R. Cai, H. W. Ng, M. Winslett, T. Z. Fu, B. Xu, X. Yang, and Z. Zhang, "Causal mechanism transfer network for time series domain adaptation in mechanical systems," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 12, no. 2, pp. 1–21, 2021.

[93] G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein, "Adversarial reprogramming of neural networks," in *International Conference on Learning Representations*, 2019.

[94] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv preprint arXiv:1808.08929*, 2018.

[95] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2021, pp. 6523–6527.

[96] P. Xiong, Y. Zhu, Z. Sun, Z. Cao, M. Wang, Y. Zheng, J. Hou, T. Huang, and Z. Que, "Application of transfer learning in continuous time series for anomaly detection in commercial aircraft flight data," in *2018 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2018, pp. 13–18.

[97] Y. Du, J. Wang, W. Feng, S. Pan, T. Qin, R. Xu, and C. Wang, "Adarnn: Adaptive learning and forecasting of time series," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 402–411.

[98] S. Baireddy, S. R. Desai, J. L. Mathieson, R. H. Foster, M. W. Chan, M. L. Comer, and E. J. Delp, "Spacecraft time-series anomaly detection using transfer learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1951–1960.

[99] Z. Shao, Z. Zhang, F. Wang, and Y. Xu, "Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1567–1577.

[100] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[101] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *Proc. Interspeech 2019*, pp. 3465–3469, 2019.

[102] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[103] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[104] X. Jiang, R. Missel, Z. Li, and L. Wang, "Sequential latent variable models for few-shot high-dimensional time-series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[105] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[106] B. Lu, X. Gan, W. Zhang, H. Yao, L. Fu, and X. Wang, "Spatio-temporal graph few-shot learning with cross-city knowledge transfer," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1162–1172.

[107] R. Wang, R. Walters, and R. Yu, "Meta-learning dynamics forecasting using task inference," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 640–21 653, 2022.

[108] T. Iwata and A. Kumagai, "Few-shot learning for time-series forecasting," *arXiv preprint arXiv:2009.14379*, 2020.

[109] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "Meta-learning framework with applications to zero-shot time-series forecasting," in *Proceedings of the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9242–9250.

[110] L. Brinkmeyer, R. R. Drumond, J. Burchert, and L. Schmidt-Thieme, "Few-shot forecasting of time-series with heterogeneous channels," *arXiv preprint arXiv:2204.03456*, 2022.

[111] A. Castellani, S. Schmitt, and B. Hammer, "Estimating the electrical power output of industrial devices with end-to-end time-series classification in the presence of label noise," in *European Conference on Machine Learning*, 2021.

[112] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[113] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of Machine Learning Research*, vol. 11, no. 12, 2010.

[114] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[115] Q. Ma, C. Chen, S. Li, and G. W. Cottrell, "Learning representations for incomplete time series clustering," in *Proceedings of the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8837–8846.

[116] P. Shi, W. Ye, and Z. Qin, "Self-supervised pre-training for time series classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[117] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," *arXiv preprint arXiv:1603.00982*, 2016.

[118] Q. Hu, R. Zhang, and Y. Zhou, "Transfer learning for short-term wind speed prediction with deep neural networks," *Renewable Energy*, vol. 85, pp. 83–95, 2016.

[119] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[120] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, "Masked feature prediction for self-supervised visual pre-training," *arXiv preprint arXiv:2112.09133*, 2021.

[121] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, "Simmim: A simple framework for masked image modeling," *arXiv preprint arXiv:2111.09886*, 2021.

[122] R. R. Chowdhury, X. Zhang, J. Shang, R. K. Gupta, and D. Hong, "Tarnet: Task-aware reconstruction for time-series transformer," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA*, 2022, pp. 14–18.

[123] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," *arXiv preprint arXiv:1905.02450*, 2019.

[124] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "Ammus: A survey of transformer-based pretrained models in natural language processing," *arXiv preprint arXiv:2108.05542*, 2021.

[125] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[126] L. Hou, Y. Geng, L. Han, H. Yang, K. Zheng, and X. Wang, "Masked token enabled pre-training: A task-agnostic approach for understanding complex traffic flow," *TechRxiv*, 2022.

[127] L. Zhao, M. Gao, and Z. Wang, "St-gsp: Spatial-temporal global semantic representation learning for urban flow prediction," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 1443–1451.

[128] I. Padhi, Y. Schiff, I. Melnyk, M. Rigotti, Y. Mroueh, P. Dognin, J. Ross, R. Nair, and E. Altman, "Tabular transformers for modeling multivariate time series," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2021, pp. 3565–3569.

[129] S. M. Shankaranarayana and D. Runje, "Attention augmented convolutional transformer for tabular time-series," in *2021 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2021, pp. 537–541.

[130] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2020, pp. 6419–6423.

[131] A. T. Liu, S.-W. Li, and H.-y. Lee, "TERA: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2351–2366, 2021.

[132] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li, "Dense contrastive learning for self-supervised visual pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3024–3033.

[133] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[134] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[135] H. Fan, F. Zhang, and Y. Gao, "Self-supervised time series representation learning by inter-intra relational reasoning," *arXiv preprint arXiv:2011.13548*, 2020.

[136] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised representation learning for time series with temporal neighborhood coding," in *International Conference on Learning Representations*, 2021.

[137] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting," *arXiv preprint arXiv:2202.01575*, 2022.

[138] 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*, 2021, pp. 3124–3135.

[139] D. Luo, W. Cheng, Y. Wang, D. Xu, J. Ni, W. Yu, X. Zhang, Y. Liu, H. Chen, and X. Zhang, "Information-aware time series meta-contrastive learning," *Submitted to International Conference on Learning Representations*, 2022.

[140] A. Hyvarinen and H. Morioka, "Unsupervised feature extraction by time-contrastive learning and nonlinear ica," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[141] H. M. Aapo Hyvarinen, "Nonlinear ica of temporally dependent stationary sources," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 460–469.

[142] L. Yang, S. Hong, and L. Zhang, "Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion," *arXiv preprint arXiv:2202.04770*, 2022.

[143] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," *arXiv preprint arXiv:2002.12478*, 2020.

[144] M. T. Nonnenmacher, L. Oldenburg, I. Steinwart, and D. Reeb, "Utilizing expert features for contrastive learning of time-series representations," in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 969–16 989.

[145] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision*, 2018, pp. 132–149.

[146] Y. Asano, C. Rupprecht, and A. Vedaldi, "Self-labelling via simultaneous clustering and representation learning," in *International Conference on Learning Representations*, 2020.

[147] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proceedings of International Conference on Learning Representations*, 2018.

[148] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[149] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive 2018," *arXiv preprint arXiv:1811.00075*, 2018.

[150] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[151] Y. B. Nikolay Laptev, Saeed Amizadeh "A benchmark dataset for time series anomaly detection," 2015, https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly.

[152] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 3009–3017.

[153] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "Ts-chief: a scalable and accurate forest algorithm for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 3, pp. 742–775, 2020.

[154] A. Dempster, D. F. Schmidt, and G. I. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 248–257.

[155] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[156] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 289–297.

[157] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[158] A. Shabani, A. Abdi, L. Meng, and T. Sylvain, "Scaleformer: iterative multi-scale refining transformers for time series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[159] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "Micn: Multi-scale local and global context modeling for long-term series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[160] Z. Wang, X. Xu, W. Zhang, G. Trajcevski, T. Zhong, and F. Zhou, "Learning latent seasonal-trend representations for time series forecasting," in *Advances in Neural Information Processing Systems*, 2022.

[161] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.

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

[163] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 187–196.

[164] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *International Conference on Learning Representations*, 2022.

[165] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[166] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[167] P. Rathore, A. Basak, S. H. Nistala, and V. Runkana, "Untargeted, targeted and universal adversarial attacks and defenses on time series," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[168] T. Wu, X. Wang, S. Qiao, X. Xian, Y. Liu, and L. Zhang, "Small perturbations are enough: Adversarial attacks on time series prediction," *Information Sciences*, vol. 587, pp. 794–812, 2022.

[169] L. Liu, Y. Park, T. N. Hoang, H. Hasson, and L. Huan, "Robust multivariate time-series forecasting: Adversarial attacks and defense mechanisms," in *The Eleventh International Conference on Learning Representations*, 2023.

[170] F. Karim, S. Majumdar, and H. Darabi, "Adversarial attacks on time series," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3309–3320, 2020.

[171] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[172] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[173] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Advances in neural information processing systems*, vol. 31, 2018.

[174] J. Li, C. Xiong, and S. C. Hoi, "Mopro: Webly supervised learning with momentum prototypes," in *International Conference on Learning Representations*, 2020.

[175] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *International Conference on Learning Representations*, 2020.

[176] X. Jiang, R. Missel, Z. Li, and L. Wang, "Sequential latent variable models for few-shot high-dimensional time-series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[177] Y. Ozyurt, S. Feuerriegel, and C. Zhang, "Contrastive learning for unsupervised domain adaptation of time series," in *The Eleventh International Conference on Learning Representations*, 2023.

[178] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," *arXiv preprint arXiv:1803.08976*, 2018.

[179] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[180] J. Narwariya, P. Malhotra, L. Vig, G. Shroff, and T. Vishnu, "Meta-learning for few-shot time series classification," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, 2020, pp. 28–36.

APPENDIX A
EXPERIMENTAL DETAILS AND RESOURCES

A.1 Source Datasets, Code and TS-PTMs

In time series domain, the statistics of publicly available benchmark datasets are shown in Table 6. Also, the statistics of existing TS-PTMs containing open-source code and the summary of TS-PTMs are shown in Table 7 and Table 8.

A.2 Experimental Datasets

In the time series domain, 128 univariate UCR time series datasets [148] and 30 multivariate UEA time series datasets [149] are commonly utilized in time-series classification studies. However, each dataset in the UCR and UEA archives does not contain the validation set employed for hyperparameter selection. For a fair analysis, we first combine the training and test sets for each dataset in UCR and UEA archives. Then, we adopt the five-fold cross-validation strategy to divide the training, validation, and test sets according to a ratio of 60%, 20%, and 20%, respectively. Finally, these repartitioned datasets are used to evaluate the time-series classification task performance. Unlike the time series forecasting task, time series classification does not require predicting past and future values of individual series. Further, each dataset in UCR and UEA archives contains several independent time series. Therefore, we can reasonably use a five-fold cross-validation strategy for time series classification. For the time-series forecasting task, like [68], we utilize ETT and Electricity datasets for the experimental analysis. In addition, following [68], we employ the Yahoo and KPI datasets for the time-series anomaly detection task.

A.3 Baselines

Transfer learning has been a significant success in the study of PTMs. Fawaz et al. [26] utilized 85 UCR datasets for time series transfer learning. Firstly, a randomly selected dataset is used as the source dataset to pre-train the Fully Convolutional Networks (FCN) [65] model via the supervised classification task. Then a randomly selected target UCR dataset is fine-tuned on the pre-trained FCN model. Meanwhile, Malhotra et al. [28] employed a symmetric RNN-based encoder-decoder architecture as the backbone. The encoder is first pre-trained using the unsupervised reconstruction task on various UCR datasets as source datasets. Then, the target dataset is fine-tuned on the pre-trained RNN-based encoder. Therefore, in terms of employing transfer learning for time series pre-training, we perform a comparative analysis utilizing a supervised classification transfer strategy and an unsupervised reconstruction transfer strategy.

Recently, TS-PTMs have achieved good performance in TSM tasks. For the time-series classification task, we select T-Loss [134], SelfTime [135], TS-TCC [38], TST [29], and TS2Vec [68] to analyze the performance of TS-PTMs and compare them with the supervised FCN [65] model. Meanwhile, we select TS2Vec [68] and CoST [137] as benchmark TS-PTMs methods for the time-series forecasting task. Also, we compare them with supervised benchmark methods in time-series forecasting, including LogTrans [157], TCN [57], Informer [14], and Autoformer [60], where LogTrans, Informer, and Autoformer are end-to-end models based on the Transformer. For the time-series anomaly detection task, we select TS2Vec [68] as the benchmark TS-PTM method. We compare it with the benchmark methods SPOT [161], DSPOT [161], LSTM-VAE [162], DONUT [163], Spectral Residual (SR) [152], and Anomaly Transformer (AT) [164] in time-series anomaly detection, where AT is a model based on the Transformer.

A.3.1 Time-Series Classification

In the time-series classification task, we first compare the performance of FCN, TCN, and Transformer for direct classification on 128 UCR time series datasets. Then, we analyze the classification performance of T-Loss, SelfTime, TS-TCC, TST, and TS2Vec after pre-training on 128 UCR and 30 UEA time series datasets. In addition, for the datasets containing missing values in the UCR and UEA archives, we use the mean-imputation method to fill the divided training, validation, and test sets, respectively. In this work, the classification accuracy of the test set is employed to evaluate classification performance. Experiments are performed on eight 1080Ti GPUs and two 3090 GPUs. For the running time of the corresponding method, we use the test results on the 3090 GPU.

The specific methods used in this study are described as follows:

FCN: Based on literature [65], we build the model using a three-layer one-dimensional **F**ully **C**onvolutional **N**etwork (**FCN**) and a one-layer global average pooling layer. We set the FCN structure and parameters based on the open source code from https://github.com/cauchyturing/UCR_Time_Series_Classification_Deep_Learning_Baseline.

TCN: Based on the open source code of literature [134], we build the **T**emporal **C**onvolutional **N**etwork (**TCN**) using Pytorch. The original TCN contains ten layers of residual blocks. To compare and analyze the three-layer structure of FCN, we employ a three-layer TCN for comparison experiments. We use the open source code from https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries for TCN structure and parameter setting.

Transformer: Zerveas et al. [29] proposed a Transformer-based model for unsupervised time series representation learning. We use the **Transformer** model in open source code from https://github.com/gzerveas/mvts_transformer for direct classification experiments on 128 UCR time series datasets.

T-Loss: Franceschi et al. [134] proposed an unsupervised time series representation learning method using TCN and a novel **T**riplet **L**oss (**T-Loss**). We use the open source code from https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries for experimental analysis.

SelfTime: Fan et al. [135] proposed a self-supervised time series representation learning framework called **SelfTime**, exploring the inter-sample and intra-temporal relation of time series for learning representations on unlabeled data. We use the open source code from https://github.com/haoyfan/SelfTime for experimental analysis.

TS-TCC: Eldele et al. [38] proposed an unsupervised **T**ime **S**eries representation learning framework via **T**emporal and **C**ontextual **C**ontrasting (**TS-TCC**), thus

TABLE 6
Benchmark Datasets in Time-Series Domain

Resource	Description	URL
UCR	128 univariate UCR time series classification datasets	https://www.cs.ucr.edu/ eamonn/time_series_data_2018/
UEA	30 multivariate UEA time series classification datasets	http://www.timeseriesclassification.com/
IHEPC	Individual Household Electric Power Consumption (IHEPC) dataset	https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption
XJTU-SU	Run-to-failure data of 15 rolling element bearings	http://biaowang.tech/xjtu-sy-bearing-datasets
MFPT	Bearing Fault Dataset	https://www.mfpt.org/fault-data-sets/
ECG Waveform	Long-term Electrocardiogram (ECG) recordings	https://archive.physionet.org/cgi-bin/atm/ATM
HAR	Human Activity Recognition Using Smartphones Data Set	https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
Sleep-EDF	Sleep Stage Classification	https://github.com/emadeldeen24/TS-TCC
Epilepsy	Epilepsy Seizure Prediction	https://github.com/emadeldeen24/TS-TCC
FD	Fault Diagnosis	https://github.com/emadeldeen24/TS-TCC
EMG	Electromyograms (EMG) measures muscle responses as electrical activity	https://github.com/mims-harvard/TFC-pretraining
ETT	A crucial indicator in the electric power long-term deployment	https://github.com/zhouhaoyi/ETDataset
Electricity	The electricity consumption (Kwh) of 321 clients	https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
Yahoo	367 hourly sampled time series with tagged anomaly points	https://webscope.sandbox.yahoo.com/
KPI	Multiple minutely sampled real KPI curves from many internet companies	https://github.com/NetManAIOps/donut/tree/master/sample_data

TABLE 7
The Open-Source Implementations of TS-PTMs.

Resource	Description	URL
TL-FCN [26]	Framework: Keras (2018 Big Data)	https://github.com/hfawaz/bigdata18
CPC [100]	Framework: PyTorch (2018 arXiv)	https://github.com/jefflai108/Contrastive-Predictive-Coding-PyTorch
T-Loss [134]	Framework: PyTorch (2019 NeurIPS)	https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries
SelfTime [135]	Framework: PyTorch (2020 arXiv)	https://github.com/haoyfan/SelfTime
TNC [136]	Framework: PyTorch (2021 ICLR)	https://github.com/sanatonek/TNC_representation_learning
Voice2Series [27]	Framework: Tensorflow (2021 ICML)	https://github.com/huckiyang/Voice2Series-Reprogramming
TS-TCC [38]	Framework: PyTorch (2021 IJCAI)	https://github.com/emadeldeen24/TS-TCC
TST [29]	Framework: PyTorch (2021 KDD)	https://github.com/gzerveas/mvts_transformer
TS2Vec [68]	Framework: PyTorch (2022 AAAI)	https://github.com/yuezhihan/ts2vec
CoST [137]	Framework: PyTorch (2022 ICLR)	https://github.com/salesforce/CoST
ExpCLR [144]	Framework: PyTorch (2022 ICML)	https://github.com/boschresearch/expclr
TF-C [31]	Framework: PyTorch (2022 NeurIPS)	https://github.com/mims-harvard/TFC-pretraining
SLVM [176]	Framework: PyTorch (2023 ICLR)	https://github.com/john-x-jiang/meta_ssm
CLUDA [177]	Framework: PyTorch (2023 ICLR)	https://github.com/oezyurty/CLUDA

learning feature representations of unlabeled time series. We use the open source code from https://github.com/emadeldeen24/TS-TCC for experimental analysis.

TST: Zerveas et al. [29] first proposed a multivariate time series unsupervised representation learning framework called **T**ime **S**eries **T**ransformer (**TST**). We use the open source code from https://github.com/gzerveas/mvts_transformer for experimental analysis.

TS2Vec: Yue et al. [68] proposed a general framework for time series representation learning in an arbitrary semantic level called **TS2Vec**. For the time series classification task, the authors employed TCN as the backbone and the low-dimensional feature representation obtained from TCN to input into an SVM classifier with RBF kernel for classification. We use the open source code from https://github.com/yuezhihan/ts2vec for experimental analysis.

A.3.2 Time-Series Forecasting

TS2Vec: Yue et al. [68] employed TCN as the backbone and the data from the last T observations to predict the observations for the next H timestamps for the time series forecasting task. We use the open source code from https://github.com/yuezhihan/ts2vec for experimental analysis.

CoST: Woo et al. [137] proposed a representation learning framework for long time series forecasting, called **CoST**, which utilizes contrastive learning to learn time series disentangled seasonal-trend representations. We use the open source code from https://github.com/salesforce/CoST for experimental analysis.

LongTrans: Li et al. [157] proposed a Transformer-based model for **Long** time series forecasting with convolutional self-attention and LogSparse **Trans**former (**LongTrans**)

to capture the time series' local context and long-term dependencies information. We use the open source code from https://github.com/mlpotter/Transformer_Time_Series for experimental analysis.

TCN: Bai et al. [57]'s extensive experiments on the sequence modeling problem indicated that a simple convolutional architecture consisting of **T**emporal **C**onvolutional **N**etworks (**TCN**) outperformed canonical recurrent networks such as LSTMs for time series forecasting on different datasets. We use the open source code from https://github.com/locuslab/TCN for experimental analysis.

Informer: Zhou et al. [14] designed an efficient transformer-based model called **Informer** for long time series forecasting. We use the open source code from https://github.com/zhouhaoyi/Informer2020 for experimental analysis.

Autoformer: Wu et al. [60] designed **Autoformer** as a novel decomposition architecture with an Auto-Correlation mechanism for long time series forecasting. We use the open source code from https://github.com/thuml/autoformer for experimental analysis.

A.3.3 Time-Series Anomaly Detection

TS2Vec: For the time-series anomaly detection task, Yue et al. [68] followed a streaming evaluation protocol [152] to perform point anomaly detection of time series. We use the open source code from https://github.com/yuezhihan/ts2vec for experimental analysis.

SPOT and DSPOT: Siffer et al. [161] proposed a time series anomaly detection method based on extreme value theory. The authors divided the proposed approach into two algorithms: **SPOT** for streaming data having any station-

TABLE 8
The summary of Time-Series Pre-Trained Models (TS-PTMs).

Year	Method	Superviesed PTMs		Unsupervised PTMs	Self-supervised PTMs		Architecture
		CLS	FORE	REC	CONSIS	PSEUDO	
2016	Audio Word2Vec [117]			✓			RNN
2016	DCFT [72]	✓					CNN&RNN
2016	TCL [140]				✓		MLP
2016	SHL-DNN [118]			✓			MLP
2017	TimeNet [28]			✓			RNN
2017	VRADA [89]	✓					RNN
2018	CPC [100]		✓		✓		CNN&RNN
2018	TL-FCN [26]	✓					CNN
2018	Encoder [63]	✓					CNN
2018	TL-CTSAD [96]		✓				RNN
2018	Speech2Vec [178]					✓	RNN
2018	TL-LSTM [37]		✓	✓			RNN
2018	HDCNN [85]	✓					CNN
2018	STL [83]	✓					CNN
2019	vq-wav2vec [114]				✓		Transformer Enc
2019	FRTL [73]	✓					CNN&RNN
2019	ConvTimeNet [56]	✓					CNN
2019	wav2vec [101]		✓		✓		CNN
2019	T-Loss [134]				✓		TCN
2020	SMS [71]	✓					CNN
2020	SelfTime [135]				✓		CNN
2020	InceptionTime [179]	✓					CNN
2020	Mockingjay [130]			✓			Transformer Enc
2020	ML-TSC [180]	✓					CNN
2020	ML-TSF [108]		✓				RNN
2020	DTL-TS [67]	✓					CNN
2020	LSTM-DANN [90]	✓					RNN
2020	CoDATS [75]	✓					CNN
2021	MLF [109]		✓				N-BEATS
2021	SASA [74]	✓					RNN
2021	SLARDA [87]	✓					Trans&CNN
2021	GCA [84]	✓					RNN
2021	AdvSKM [88]	✓					CNN
2021	CALDA [91]	✓					CNN
2021	CMTN [92]	✓					RNN
2021	TSAD-TL [98]		✓				RNN
2021	TNC [136]				✓		RNN/TCN
2021	Voice2Series [27]	✓					Trans&CNN&RNN
2021	SleepPriorCL [36]				✓	✓	CNN
2021	TS-TCC [38]		✓		✓		CNN/Transformer
2021	TS-Transformer [29]			✓			Transformer Enc
2021	SSL-TS [116]			✓	✓		Transformer Enc
2021	InfoTS [139]				✓		TCN
2021	TabBERT [128]			✓			Transformer Enc
2021	TabAConvBERT [129]			✓			Transformer Enc
2021	TERA [131]			✓			Transformer Enc
2021	AdaRNN [97]		✓				RNN
2022	TSSN [126]			✓			Transformer Enc
2022	ST-GSP [127]			✓			Transformer Enc
2022	CoST [137]				✓		TCN
2022	TS2Vec [68]				✓		TCN
2022	BTSF [142]				✓		TCN
2022	STEP [99]		✓				CNN
2022	TARNet [122]			✓			Transformer
2022	ST-GFSL [106]		✓				RNN
2022	ExpCLR [144]				✓		TCN
2022	TimeHetNet [110]		✓				CNN&RNN
2022	CRT [32]	✓					Transformer Enc
2022	SPGN [20]	✓					CNN
2022	TF-C [31]				✓		CNN
2023	SLVM [176]		✓				RNN
2023	CLUDA [177]				✓		TCN

ary distribution, and **DSPOT** for streaming data that can be subject to concept drift. We use the open source code from https://github.com/Amossys-team/SPOT for experimental analysis.

LSTM-VAE: Park et al. [162] proposed a **L**ong **S**hort-**T**erm **M**emory-based **V**ariational **a**uto**E**ncoder (**LSTM-VAE**) model for outlier detection of multimodal sensory signals. We use the open source code from https://github.com/SchindlerLiang/VAE-for-Anomaly-Detection for experimental analysis.

DONUT: Xu et al. [163] proposed an unsupervised anomaly detection algorithm based on a variational autoencoder called **DONUT**. We use the open source code from https://github.com/NetManAIOps/donut for experimental analysis.

SR: Ren et al. [152] proposed an algorithm based on **S**pectral **R**esidual (**SR**) and CNN for anomaly detection of time series. Since the authors do not provide the open source code, we use the experimental results on Yahoo and KPI datasets in the original paper for comparative analysis.

AT: Xu et al. [164] proposed **A**nomaly **T**ransformer (**AT**) with a new anomaly-attention mechanism for unsupervised detection of anomaly points in time series. We use the open source code from https://github.com/spencerbraun/anomaly_transformer_pytorch for experimental analysis.

For the above baselines in three major time-series mining tasks (classification, forecasting, and anomaly detection), we use uniform random seeds for the model's training. In addition, the dataset partitioning used by all baselines is kept consistent. Further, the hyperparameter settings of all baselines are set according to the parameters provided by the original authors, and the details can be found in our open-source code https://github.com/qianlima-lab/time-series-ptms.

A.4 Implementation Details

For the time-series classification task, we normalize each series of the UCR and UEA datasets via z-score [39], and build FCN and TCN models using Pytorch according to the settings of [39], [134]. Adam is adopted as the optimizer with a learning rate of 0.001 and a maximum batch size of 128. The maximum epoch of the source UCR times series dataset for transfer learning is 2000, while the maximum epoch of the target UCR times series dataset is 1000. Also, a uniform early stopping rule is used for all baselines in training. The maximum epoch of the source dataset of four independent time series scenarios (refer to Table 11) is 40, while the maximum epoch of the corresponding target dataset is 100. For the classification accuracy of the target UCR time series, we use the average test accuracy on five-fold test sets by running one seed. For the classification accuracy of the target datasets on four independent time series scenarios, we use the average test accuracy on the test set by running ten different seeds. For the time-series forecasting and anomaly detection tasks, we follow the way of [68] to preprocess the datasets and set the hyperparameters. In addition, the comparative benchmark methods for time-series classification, forecasting, and anomaly detection tasks are reproduced and analyzed using the open-source codes provided by the original authors. To ensure the reproducibility

TABLE 9
Details of the source UCR time series datasets used for transfer learning.

ID	Dataset Name	Total Sample Size	Length	Class
1	Crop	24000	46	24
2	ElectricDevices	16637	96	7
3	StarLightCurves	9236	1024	3
4	Wafer	7164	152	2
5	ECG5000	5000	140	5
6	TwoPatterns	5000	128	4
7	FordA	4921	500	2
8	UWaveGestureLibraryAll	4478	945	8
9	UWaveGestureLibraryX	4478	315	8
10	UWaveGestureLibraryY	4478	315	8
11	UWaveGestureLibraryZ	4478	315	8
12	FordB	4446	500	2
13	ChlorineConcentration	4307	166	3
14	NonInvasiveFetalECGThorax1	3765	750	42
15	NonInvasiveFetalECGThorax2	3765	750	42

of the experimental results, we use a uniform random seed for all baselines. Finally, all experiments are done on eight 1080Ti GPUs and two 3090 GPUs with Ubuntu 18.04 system. Also, the training times in the main experimental results are all run on 3090 GPUs.

APPENDIX B
FULL RESULTS

In this section, we give detailed classification results based on time series PTMs on 128 UCR and 30 UEA datasets.

B.1 Comparison of Transfer Learning PTMs based on Supervised Classification and Unsupervised Reconstruction

The test supervised classification accuracy of FCN, TCN, and Transformer on 128 UCR time series datasets is given in Table 12. To facilitate the layout and reading of the test classification results, the standard deviation of the classification accuracy for each dataset is not given in Table 12. For the datasets used in transfer learning PTMs, please refer to Tables 9, 10, and 11.

B.2 Comparison of PTMs based on Transfomer and Consistency

The pre-training test classification accuracy using transformer-based and contrastive learning on 128 UCR and 30 UEA time series datasets are shown in Tables 13 and 14. Also, for the convenience of layout and reading of the test classification results, the standard deviations of the classification performance evaluated for each dataset are not given in Tables 13 and 14.

Fig. 14. Classification comparison results based on transfer learning. The values indicate the difference in classification accuracy on the FCN encoder with and without transfer learning. A value greater than zero indicates positive transfer, while a value less than zero indicates negative transfer. Red background is for positive values and blue for negative, with larger magnitude values more saturated.

TABLE 10
Details of the target UCR time series datasets used for transfer
learning.

	15 Minmum Sample Size Target Datasets			
ID	Dataset Name	Total Sample Size	Length	Class
1	BirdChicken	40	512	2
2	BeetleFly	40	512	2
3	Coffee	56	286	2
4	OliveOil	60	570	4
5	Beef	60	470	5
6	Rock	70	2844	4
7	ShakeGestureWiimoteZ	100	385	10
8	PickupGestureWiimoteZ	100	361	10
9	Wine	111	234	2
10	FaceFour	112	350	4
11	Meat	120	448	3
12	Car	120	577	4
13	Lightning2	121	637	2
14	Herring	128	512	2
15	Lightning7	143	319	7
	15 Medium Sample Size Target Datasets			
ID	Dataset Name	Total Sample Size	Length	Class
1	Earthquakes	461	512	2
2	Haptics	463	1092	5
3	Computers	500	720	2
4	DistalPhalanxTW	539	80	6
5	DistalPhalanxOutlineAgeGroup	539	80	3
6	MiddlePhalanxTW	553	80	6
7	MiddlePhalanxOutlineAgeGroup	554	80	3
8	SyntheticControl	600	60	6
9	ProximalPhalanxTW	605	80	6
10	ProximalPhalanxOutlineAgeGroup	605	80	3
11	SonyAIBORobotSurface1	621	70	2
12	InlineSkate	650	1882	7
13	EOGHorizontalSignal	724	1250	12
14	EOGVerticalSignal	724	1250	12
15	SmallKitchenAppliances	750	720	3
	15 Maxmum Sample Size Target Datasets			
ID	Dataset Name	Total Sample Size	Length	Class
1	MoteStrain	1272	84	2
2	HandOutlines	1370	2709	2
3	CinCECGTorso	1420	1639	4
4	Phoneme	2110	1024	39
5	InsectWingbeatSound	2200	256	11
6	FacesUCR	2250	131	14
7	FaceAll	2250	131	14
8	Mallat	2400	1024	8
9	MixedShapesSmallTrain	2525	1024	5
10	PhalangesOutlinesCorrect	2658	80	2
11	FreezerSmallTrain	2878	301	2
12	MixedShapesRegularTrain	2925	1024	5
13	FreezerRegularTrain	3000	301	2
14	Yoga	3300	426	2
15	MelbournePedestrian	3633	24	10

TABLE 11
The detailed information of four independent time series scenarios datasets. For column *Samples*, *Train* is the training set, *Val* is the validation set, and *Test* is the test set.

Scenario	Transfer Learning	Dataset	# Samples	# Channels	# Classes	# Length
Neurological Stage Detection	Pre-training (Source)	SleepEEG	Train (371055)	1	5	200
	Fine-turning (Target)	Epilepsy	Train (60), Val (20), Test (11420)	1	2	178
Mechanical Device Diagnosis	Pre-training (Source)	FD-A	Train (8184)	1	3	5120
	Fine-turning (Target)	FD-B	Train (60), Val (21), Test (13559)	1	3	5120
Activity Recognition	Pre-training (Source)	HAR	Train (10299)	9	6	128
	Fine-turning (Target)	Gesture	Train (320), Val (120), Test (120)	3	8	315
Physical Status Monitoring	Pre-training (Source)	ECG	Train (43673)	1	4	1500
	Fine-turning (Target)	EMG	Train (122), Val (41), Test (41)	1	3	1500

TABLE 12
Test classification accuracy using FCN, TCN, and Transformer for directly supervised classification training on 128 UCR datasets.

ID	Dataset	FCN	TCN (three layers)	TCN (ten layers)	Transformer
1	ACSF1	0.6250	0.3950	0.5400	0.3450
2	Adiac	0.6492	0.2549	0.2894	0.6415
3	AllGestureWiimoteX	0.7110	0.2370	0.6630	0.3720
4	AllGestureWiimoteY	0.6740	0.4980	0.6960	0.4950
5	AllGestureWiimoteZ	0.7200	0.7050	0.6300	0.3380
6	ArrowHead	0.8958	0.3838	0.3838	0.8676
7	Beef	0.6000	0.3167	0.4167	0.6500
8	BeetleFly	0.8250	0.6500	0.7000	0.8500
9	BirdChicken	0.9250	0.6250	0.8500	0.9944
10	BME	0.7833	0.9944	1.0000	0.7167
11	Car	0.9417	0.8750	0.9417	1.0000
12	CBF	1.0000	1.0000	1.0000	0.7250
13	Chinatown	0.9753	0.9808	0.9836	0.9836
14	ChlorineConcentration	0.9984	0.5356	0.5407	0.9970
15	CinCECGTorso	0.9986	0.9915	0.9500	0.9951
16	Coffee	1.0000	1.0000	1.0000	1.0000
17	Computers	0.8800	0.8740	0.5000	0.6680
18	CricketX	0.9064	0.7615	0.6974	0.5513
19	CricketY	0.9000	0.6551	0.7013	0.5526
20	CricketZ	0.8833	0.8026	0.7679	0.5641
21	Crop	0.1312	0.0417	0.0417	0.7239
22	DiatomSizeReduction	0.9969	0.8913	0.6569	0.9969
23	DistalPhalanxOutlineAgeGroup	0.9091	0.5974	0.8032	0.8107
24	DistalPhalanxOutlineCorrect	0.8917	0.9020	0.8962	0.8185
25	DistalPhalanxTW	0.8423	0.4731	0.4731	0.7792
26	DodgerLoopDay	0.7486	0.7990	0.2833	0.4869
27	DodgerLoopGame	0.8167	0.5190	0.5190	0.8798
28	DodgerLoopWeekend	0.9812	0.7089	0.7089	0.9871
29	Earthquakes	0.8482	0.7983	0.7983	0.5995
30	ECG200	0.9350	0.9400	0.6650	0.7374
31	ECG5000	0.9258	0.5838	0.5380	0.8750
32	ECGFiveDays	1.0000	1.0000	1.0000	0.9494
33	ElectricDevices	0.6731	0.1845	0.2633	0.5428
34	EOGHorizontalSignal	0.7376	0.5347	0.7336	1.0000
35	EOGVerticalSignal	0.6519	0.7003	0.7514	0.8053
36	EthanolLevel	0.8307	0.3717	0.9154	0.7720
37	FaceAll	0.9462	0.9911	0.9711	0.9556
38	FaceFour	0.9557	0.9913	0.9648	0.9296
39	FacesUCR	0.9969	0.9893	0.9876	0.9618
40	FiftyWords	0.5072	0.2155	0.4243	0.6740
41	Fish	0.9686	0.4086	0.9400	0.8086
42	FordA	0.9734	0.9651	0.9705	0.5237
43	FordB	0.9312	0.9517	0.9559	0.5461
44	FreezerRegularTrain	0.9790	0.5000	0.5000	0.9987
45	FreezerSmallTrain	0.8359	0.5000	0.5000	0.9997
46	Fungi	0.9118	0.3987	0.0980	1.0000
47	GestureMidAirD1	0.6954	0.2781	0.4621	0.5087
48	GestureMidAirD2	0.5860	0.2930	0.4650	0.4972
49	GestureMidAirD3	0.3819	0.2486	0.3407	0.2694
50	GesturePebbleZ1	0.9541	0.5330	0.8852	0.6875
51	GesturePebbleZ2	0.9343	0.3584	0.3185	0.6615
52	GunPoint	1.0000	0.9950	0.5900	0.9900
53	GunPointAgeSpan	0.9956	0.9956	0.9956	0.9822
54	GunPointMaleVersusFemale	0.9978	0.9956	0.9978	1.0000
55	GunPointOldVersusYoung	0.9934	0.9823	0.9557	0.9756
56	Ham	0.6632	0.8828	0.9442	0.8750
57	HandOutlines	0.8956	0.6752	0.6387	0.8757
58	Haptics	0.6158	0.5623	0.7261	0.4083
59	Herring	0.6406	0.5938	0.6092	0.6243
60	HouseTwenty	0.9875	0.5597	0.5597	0.7292
61	InlineSkate	0.7046	0.4923	0.8169	0.3369
62	InsectEPGRegularTrain	0.9935	0.8232	0.9457	0.7651
63	InsectEPGSmallTrain	0.8761	0.8270	0.6836	0.7971
64	InsectWingbeatSound	0.7309	0.7527	0.7364	0.6691
65	ItalyPowerDemand	0.9891	0.9781	0.9754	0.9735
66	LargeKitchenAppliances	0.9507	0.7680	0.8707	0.5187
67	Lightning2	0.7940	0.6113	0.8843	0.7593
68	Lightning7	0.8406	0.2655	0.8123	0.6352
69	Mallat	0.9975	0.9858	0.8213	0.9800
70	Meat	0.9667	0.4417	0.3917	0.9917
71	MedicalImages	0.8397	0.5206	0.5206	0.7388
72	MelbournePedestrian	0.8145	0.8748	0.9219	0.8792
73	MiddlePhalanxOutlineAgeGroup	0.8177	0.7635	0.7274	0.7383
74	MiddlePhalanxOutlineCorrect	0.8834	0.9372	0.9092	0.8451
75	MiddlePhalanxTW	0.7074	0.5967	0.5967	0.6184
76	MixedShapesRegularTrain	0.7463	0.2010	0.1908	0.8732
77	MixedShapesSmallTrain	0.7339	0.1893	0.1893	0.8745
78	MoteStrain	0.9819	0.9811	0.9772	0.9584
79	NonInvasiveFetalECGThorax1	0.8542	0.3057	0.4167	0.8510
80	NonInvasiveFetalECGThorax2	0.8093	0.5216	0.4733	0.8951
81	OliveOil	0.7000	0.5000	0.4667	0.4955
82	OSULeaf	0.9887	0.6668	0.9167	0.8667
83	PhalangesOutlinesCorrect	0.8691	0.9079	0.9090	0.1469
84	Phoneme	0.3019	0.1128	0.1128	0.5600
85	PickupGestureWiimoteZ	0.7800	0.6300	0.6700	0.0352
86	PigAirwayPressure	0.1091	0.0706	0.0351	0.1345
87	PigArtPressure	0.5164	0.1251	0.0446	0.0607
88	PigCVP	0.4822	0.1764	0.0446	0.5708
89	PLAID	0.4348	0.4004	0.4321	0.8295
90	Plane	1.0000	1.0000	0.6524	0.9857
91	PowerCons	0.9139	0.9222	0.9694	0.9528
92	ProximalPhalanxOutlineAgeGroup	0.8727	0.7818	0.7884	0.8496
93	ProximalPhalanxOutlineCorrect	0.9214	0.9181	0.9237	0.8833
94	ProximalPhalanxTW	0.8083	0.7074	0.4165	0.8182
95	RefrigerationDevices	0.5747	0.7307	0.7640	0.4013
96	Rock	0.6429	0.7000	0.3714	0.6571
97	ScreenType	0.6093	0.7573	0.7240	0.4373
98	SemgHandGenderCh2	0.9378	0.6111	0.6000	0.8422
99	SemgHandMovementCh2	0.7156	0.6444	0.5711	0.4122
100	SemgHandSubjectCh2	0.6922	0.3889	0.7744	0.7567
101	ShakeGestureWiimoteZ	0.9200	0.3700	0.6600	0.5300
102	ShapeletSim	0.9700	0.5000	0.5000	0.4550
103	ShapesAll	0.7267	0.2750	0.5250	0.6675
104	SmallKitchenAppliances	0.7987	0.3480	0.3333	0.5320
105	SmoothSubspace	0.9467	0.9433	0.9600	0.9800
106	SonyAIBORobotSurface1	0.9984	0.9952	0.9888	0.9952
107	SonyAIBORobotSurface2	0.9990	0.9980	0.9908	0.9643
108	StarLightCurves	0.9803	0.8555	0.6883	0.9153
109	Strawberry	0.9756	0.9705	0.9654	0.9613
110	SwedishLeaf	0.9929	0.9040	0.7884	0.8880
111	Symbols	0.9961	0.8206	0.9843	0.9529
112	SyntheticControl	0.9700	0.5000	0.8283	0.9700
113	ToeSegmentation1	0.9664	0.9518	0.5628	0.6864
114	ToeSegmentation2	0.9282	0.7471	0.7471	0.8316
115	Trace	1.0000	1.0000	1.0000	1.0000
116	TwoLeadECG	1.0000	0.9991	0.9991	1.0000
117	TwoPatterns	0.9454	0.6958	0.5568	0.9862
118	UMD	0.8778	0.3333	0.9944	1.0000
119	UWaveGestureLibraryAll	0.9245	0.9359	0.9884	0.8957
120	UWaveGestureLibraryX	0.8665	0.8761	0.9134	0.7347
121	UWaveGestureLibraryY	0.7906	0.8091	0.8627	0.6876
122	UWaveGestureLibraryZ	0.8484	0.8569	0.8629	0.6648
123	Wafer	1.0000	0.8936	0.8936	0.9987
124	Wine	0.5411	0.5316	0.5229	0.7012
125	WordSynonyms	0.5967	0.2210	0.2210	0.6243
126	Worms	0.7906	0.4225	0.4225	0.4028
127	WormsTwoClass	0.8025	0.8528	0.5968	0.5504
128	Yoga	0.9718	0.9645	0.9788	0.8906
	Avg Acc	**0.8296**	0.6610	0.6835	0.7415
	Avg Rank	**1.68**	2.74	2.64	2.56
	P-value	-	1.38E-16	1.44E-13	5.41E-07
	Training Time (hours)	4.87	**3.06**	10.99	263.48

TABLE 13
Test classification accuracy using Transformer-based and consistency-based PTMs on 128 UCR datasets.

ID	Dataset	Supervised (FCN)	T-Loss	Selftime	TS-TCC	TST	TS2Vec
1	ACSF1	0.6250	0.8050	0.7250	0.5700	0.7050	0.8800
2	Adiac	0.6492	0.7811	0.7322	0.4003	0.7363	0.8066
3	AllGestureWiimoteX	0.7110	0.7170	0.6461	0.7201	0.5280	0.7820
4	AllGestureWiimoteY	0.6740	0.8110	0.7170	0.7615	0.3850	0.8400
5	AllGestureWiimoteZ	0.7200	0.7470	0.6485	0.6567	0.4780	0.7730
6	ArrowHead	0.8958	0.8770	0.8155	0.8200	0.8722	0.8908
7	BME	0.6000	0.5333	0.9722	0.9889	0.9722	0.9944
8	Beef	0.8250	0.8750	0.5167	0.6333	0.6667	0.6833
9	BeetleFly	0.9250	0.9750	0.8000	0.6250	0.6750	0.9250
10	BirdChicken	0.7833	0.9833	0.9000	0.6750	0.8500	0.9750
11	CBF	0.9417	0.8250	0.9967	0.9967	0.9946	1.0000
12	Car	1.0000	0.8667	0.6583	0.7167	0.7583	0.8833
13	Chinatown	0.9753	0.9807	0.9669	0.9507	0.9699	0.9726
14	ChlorineConcentration	0.9984	0.9988	0.6116	0.8424	0.9774	0.9998
15	CinCECGTorso	0.9986	0.9944	0.9877	0.9995	0.9993	0.9972
16	Coffee	1.0000	0.9818	0.9273	0.9455	0.8758	1.0000
17	Computers	0.8800	0.7140	0.7920	0.6080	0.6880	0.6940
18	CricketX	0.9064	0.7795	0.7316	0.7327	0.7026	0.8321
19	CricketY	0.9000	0.7487	0.6904	0.7161	0.6962	0.8256
20	CricketZ	0.8833	0.7833	0.7411	0.7384	0.6936	0.8487
21	Crop	0.1312	0.7063	0.6634	0.7801	0.7709	0.7448
22	DiatomSizeReduction	0.9969	0.9969	0.9723	0.9783	1.0000	0.9969
23	DistalPhalanxOutlineAgeGroup	0.9091	0.8274	0.7959	0.8200	0.8404	0.8125
24	DistalPhalanxOutlineCorrect	0.8917	0.8471	0.7895	0.7646	0.8254	0.8185
25	DistalPhalanxTW	0.8423	0.7421	0.7514	0.7755	0.7551	0.7792
26	DodgerLoopDay	0.7486	0.4819	0.4742	0.5885	0.5377	0.6391
27	DodgerLoopGame	0.8167	0.8921	0.8282	0.8857	0.8988	0.9563
28	DodgerLoopWeekend	0.9812	0.9677	0.9681	0.9742	0.9679	0.9812
29	ECG200	0.8482	0.7983	0.7900	0.7950	0.8600	0.8800
30	ECG5000	0.9350	0.8500	0.9317	0.9525	0.9502	0.9528
31	ECGFiveDays	0.9258	0.9478	0.9873	0.9972	0.9977	1.0000
32	EOGHorizontalSignal	1.0000	1.0000	0.7579	0.6295	0.6781	0.7279
33	EOGVerticalSignal	0.6731	0.8736	0.6774	0.6190	0.4806	0.6519
34	Earthquakes	0.7376	0.7417	0.7983	0.7505	0.7961	0.7983
35	ElectricDevices	0.6519	0.6547	0.7173	0.8639	0.8758	0.8848
36	EthanolLevel	0.8307	0.5627	0.4864	0.2985	0.7919	0.5897
37	FaceAll	0.9462	0.7720	0.9552	0.9914	0.9720	0.9563
38	FaceFour	0.9557	0.9281	0.8830	0.9379	0.9375	0.9640
39	FacesUCR	0.9969	0.9729	0.9544	0.9899	0.9702	0.9902
40	FiftyWords	0.5072	0.8088	0.6705	0.7628	0.7381	0.8022
41	Fish	0.9686	0.8743	0.8514	0.8114	0.8628	0.9343
42	FordA	0.9734	0.9114	0.8787	0.9342	0.8990	0.9303
43	FordB	0.9312	0.9035	0.8771	0.9087	0.8650	0.9132
44	FreezerRegularTrain	0.9790	0.9980	0.9974	0.9969	0.9987	0.9977
45	FreezerSmallTrain	0.8359	0.9965	0.9975	0.9966	0.9979	0.9969
46	Fungi	0.9118	1.0000	0.9316	0.9557	0.9902	1.0000
47	GestureMidAirD1	0.6954	0.5919	0.7067	0.6120	0.6181	0.6479
48	GestureMidAirD2	0.5860	0.4973	0.5504	0.4853	0.5475	0.5270
49	GestureMidAirD3	0.3819	0.3108	0.3664	0.3137	0.3462	0.3315
50	GesturePebbleZ1	0.9541	0.6420	0.9043	0.8977	0.7761	0.9507
51	GesturePebbleZ2	0.9343	0.6614	0.8817	0.8914	0.8159	0.9473
52	GunPoint	1.0000	0.9850	0.9550	0.9500	0.9700	0.9950
53	GunPointAgeSpan	0.9956	0.9756	0.9800	0.9578	0.9778	0.9889
54	GunPointMaleVersusFemale	0.9978	0.9911	0.9845	0.9823	0.9867	0.9956
55	GunPointOldVersusYoung	0.9934	0.9468	0.9490	0.9512	0.9667	0.9956
56	Ham	0.6632	0.8080	0.7193	0.8275	0.8450	0.8741
57	HandOutlines	0.8956	0.9073	0.8425	0.8657	0.7170	0.9168
58	Haptics	0.6158	0.4879	0.4730	0.4990	0.4794	0.5594
59	Herring	0.6406	0.5929	0.6086	0.5452	0.6400	0.6806
60	HouseTwenty	0.9875	0.8935	0.9746	0.8244	0.7673	0.9496
61	InlineSkate	0.7046	0.6354	0.4202	0.4172	0.4754	0.6415
62	InsectEPGRegularTrain	0.9935	0.9839	0.9616	0.8554	0.8327	0.9807
63	InsectEPGSmallTrain	0.8761	0.9850	0.9774	0.8308	0.8235	0.9699
64	InsectWingbeatSound	0.7309	0.6673	0.6092	0.7119	0.6982	0.7077
65	ItalyPowerDemand	0.9891	0.9671	0.9674	0.9660	0.9726	0.9745
66	LargeKitchenAppliances	0.9507	0.8907	0.9403	0.7401	0.5693	0.9147
67	Lightning2	0.7940	0.9090	0.7860	0.7187	0.6787	0.9010
68	Lightning7	0.8406	0.8596	0.7419	0.7419	0.7059	0.8389
69	Mallat	0.9975	0.9875	0.9901	0.9857	0.9954	0.9962
70	Meat	0.9667	1.0000	0.8917	0.5417	0.9833	1.0000
71	MedicalImages	0.8397	0.8291	0.7530	0.8175	0.8010	0.8431
72	MelbournePedestrian	0.8145	0.8489	0.8485	0.9141	0.5888	0.8986
73	MiddlePhalanxOutlineAgeGroup	0.8177	0.7544	0.7527	0.7563	0.7600	0.7347
74	MiddlePhalanxOutlineCorrect	0.8834	0.8294	0.7936	0.7838	0.8507	0.8507
75	MiddlePhalanxTW	0.7074	0.6202	0.6184	0.6256	0.6456	0.6455
76	MixedShapesRegularTrain	0.7463	0.9586	0.9580	0.9310	0.9296	0.9450
77	MixedShapesSmallTrain	0.7339	0.9541	0.9530	0.9255	0.9275	0.9410
78	MoteStrain	0.9819	0.9670	0.9505	0.9435	0.9772	0.9717
79	NonInvasiveFetalECGThorax1	0.8542	0.9384	0.8799	0.9253	0.9129	0.9392
80	NonInvasiveFetalECGThorax2	0.8093	0.9461	0.9012	0.9400	0.9286	0.9490
81	OSULeaf	0.9887	0.8304	0.5333	0.4000	0.5334	0.8500
82	OliveOil	0.7000	0.8833	0.8370	0.6493	0.6111	0.8937
83	PLAID	0.4348	0.8424	0.3912	0.4832	0.7421	0.5503
84	PhalangesOutlinesCorrect	0.8691	0.3664	0.7466	0.7803	0.8423	0.8439
85	Phoneme	0.3019	0.7200	0.4144	0.3587	0.1929	0.4204
86	PickupGestureWiimoteZ	0.7800	0.1700	0.8000	0.7800	0.6200	0.8400
87	PigAirwayPressure	0.1091	0.6446	0.2182	0.0575	0.0353	0.4038
88	PigArtPressure	0.5164	0.6731	0.9456	0.0929	0.1312	0.9424
89	PigCVP	0.4822	0.5121	0.7083	0.0449	0.0321	0.8753
90	Plane	1.0000	1.0000	0.9762	0.9714	0.9381	0.9905
91	PowerCons	0.9139	0.9194	0.8889	0.9444	0.9722	0.9861
92	ProximalPhalanxOutlineAgeGroup	0.8727	0.8496	0.8446	0.8231	0.8496	0.8545
93	ProximalPhalanxOutlineCorrect	0.9214	0.8889	0.8383	0.8303	0.8799	0.8833
94	ProximalPhalanxTW	0.8083	0.8215	0.8000	0.7653	0.8347	0.8281
95	RefrigerationDevices	0.5747	0.7773	0.7193	0.5170	0.5146	0.7627
96	Rock	0.6429	0.7143	0.6857	0.7857	0.7714	0.8286
97	ScreenType	0.6093	0.5480	0.6852	0.4805	0.4827	0.5520
98	SemgHandGenderCh2	0.9378	0.7456	0.8745	0.9516	0.9111	0.9544
99	SemgHandMovementCh2	0.7156	0.4056	0.5769	0.6980	0.4667	0.7767
100	SemgHandSubjectCh2	0.6922	0.6356	0.8286	0.9310	0.8467	0.9344
101	ShakeGestureWiimoteZ	0.9200	0.9100	0.8900	0.8500	0.6800	0.9300
102	ShapeletSim	0.9700	0.9200	0.9450	0.6050	0.5600	0.9850
103	ShapesAll	0.7267	0.8925	0.8835	0.7777	0.7292	0.9133
104	SmallKitchenAppliances	0.7987	0.7227	0.8256	0.7095	0.5747	0.7347
105	SmoothSubspace	0.9467	0.9233	0.9367	0.9433	0.9800	0.9633
106	SonyAIBORobotSurface1	0.9984	0.9920	0.9726	0.9775	0.9823	0.9952
107	SonyAIBORobotSurface2	0.9990	0.9878	0.9376	0.9849	0.9847	0.9949
108	StarLightCurves	0.9803	0.9783	0.9793	0.9738	0.9734	0.9801
109	Strawberry	0.9756	0.9614	0.9426	0.9414	0.9735	0.9685
110	SwedishLeaf	0.9929	0.9404	0.9158	0.9423	0.9253	0.9632
111	Symbols	0.9961	0.9824	0.9733	0.9765	0.9863	0.9765
112	SyntheticControl	0.9700	0.9883	0.9683	0.9950	0.9767	0.9983
113	ToeSegmentation1	0.9664	0.9551	0.9143	0.9181	0.6903	0.9588
114	ToeSegmentation2	0.9282	0.9219	0.9283	0.8250	0.8435	0.9517
115	Trace	1.0000	1.0000	0.9950	0.9150	0.9700	1.0000
116	TwoLeadECG	1.0000	0.9983	0.9932	0.9965	0.9914	0.9991
117	TwoPatterns	0.9454	1.0000	0.9513	0.9990	0.9994	1.0000
118	UMD	0.8778	0.9944	0.9389	0.9611	0.9778	0.9944
119	UWaveGestureLibraryAll	0.9245	0.9518	0.8595	0.9750	0.9602	0.9652
120	UWaveGestureLibraryX	0.8665	0.8457	0.6680	0.8385	0.8091	0.8513
121	UWaveGestureLibraryY	0.7906	0.7657	0.5342	0.7573	0.7300	0.7874
122	UWaveGestureLibraryZ	0.8484	0.8021	0.6076	0.7762	0.4790	0.8024
123	Wafer	1.0000	0.9989	0.9587	0.9982	0.9992	0.9990
124	Wine	0.5411	0.9553	0.6743	0.5221	0.5854	0.9648
125	WordSynonyms	0.5967	0.8022	0.5718	0.7444	0.7083	0.7989
126	Worms	0.7906	0.6821	0.6860	0.5423	0.4148	0.7210
127	WormsTwoClass	0.8025	0.7634	0.7557	0.6898	0.6240	0.7716
128	Yoga	0.9718	0.9661	0.7699	0.9207	0.9497	0.9709
	Avg Acc	0.8296	0.8325	0.8017	0.7807	0.7755	**0.8691**
	Avg Rank	2.73	3.28	4.38	4.30	4.15	**2.07**

TABLE 14
Test classification accuracy using Transformer-based and consistency-based PTMs on 30 UEA datasets

ID	Dataset	Supervised (FCN)	T-Loss	TS-TCC	TST	TS2Vec
1	ArticularyWordRecognition	**0.9983**	0.8974	0.9757	0.9826	0.9913
2	AtrialFibrillation	0.3667	0.4000	0.3667	**0.5667**	0.1667
3	BasicMotions	**1.0000**	0.6375	0.9625	0.9000	0.9500
4	CharacterTrajectories	0.9738	0.9598	0.9882	0.9815	**0.9941**
5	Cricket	0.9500	0.6778	0.9278	0.9611	**0.9889**
6	DuckDuckGeese	**0.7200**	0.2500	0.2800	0.5100	0.4000
7	EigenWorms	0.8839	0.5056	0.7476	0.5791	**0.8996**
8	Epilepsy	**0.9855**	0.5600	0.9236	0.7018	0.9818
9	ERing	**0.9867**	0.9200	0.9667	0.9533	0.9600
10	EthanolConcentration	**0.3321**	0.2500	0.2633	0.2596	0.2157
11	FaceDetection	**0.7987**	0.5255	0.6973	0.6935	0.5228
12	FingerMovements	**0.6638**	0.5145	0.5047	0.5265	0.5239
13	HandMovementDirection	**0.6550**	0.2992	0.4144	0.4660	0.3079
14	Handwriting	**0.9520**	0.3480	0.7266	0.3080	0.8740
15	Heartbeat	**0.8363**	0.7213	0.7066	0.7164	0.7115
16	InsectWingbeat	0.1503	0.2676	0.6004	**0.6253**	0.2805
17	JapaneseVowels	**0.9937**	0.7406	0.9594	0.9609	0.9656
18	Libras	**0.9250**	0.7306	0.7444	0.7972	0.8500
19	LSST	0.6016	0.3155	0.5682	0.5127	**0.6268**
20	MotorImagery	**0.7572**	0.5025	0.5318	0.5157	0.5267
21	NATOPS	**0.9278**	0.5944	0.7806	0.8722	0.8417
22	PEMS-SF	**0.9318**	0.6864	0.8159	0.8068	0.8682
23	PenDigits	**0.9979**	0.9933	0.9932	0.9895	0.9953
24	PhonemeSpectra	0.1570	0.0964	0.2356	0.0562	**0.2478**
25	RacketSports	**0.9408**	0.6169	0.8317	0.8181	0.8709
26	SelfRegulationSCP1	**0.8486**	0.8128	0.8288	0.7968	0.7736
27	SelfRegulationSCP2	**0.6395**	0.5000	0.5105	0.4868	0.5132
28	SpokenArabicDigits	0.8131	0.9519	0.9960	0.9849	**0.9975**
29	StandWalkJump	0.4067	0.4533	0.4000	**0.5133**	0.5067
30	UWaveGestureLibrary	**0.9614**	0.8591	0.9273	0.9205	0.9500
	Avg Acc	**0.7718**	0.5863	0.7059	0.6921	0.7101
	Avg Rank	**1.73**	4.33	3.10	3.23	2.57