Information Sciences 624 (2023) 147–164

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Unsupervised multimodal domain adversarial network for time series classification ☆

Liang Xi *, Yujia Liang, Xunhua Huang, Han Liu, Ao Li

Harbin University of Science and Technology, 52 Xuefu Road, Harbin 150080, China

ARTICLE INFO

Article history:
Received 25 September 2022
Received in revised form 6 November 2022
Accepted 25 December 2022
Available online 30 December 2022

Keywords:
Time series classification
Transfer learning
Unsupervised domain adaption
Multimodal
Domain adversarial network
Joint maximum mean discrepancy

ABSTRACT

Unsupervised Domain Adaptation (UDA) is an ideal transfer learning method, which can use labeled source data to improve the classification performance of unlabeled target data. At present, the UDA methods for Time Series Classification (TSC) only use time-domain data or frequency-domain data as the input, and ignore fusing them, resulting in insufficient feature extraction and inaccurate source-target distribution alignment. Therefore, we propose an unsupervised Multimodal Domain Adversarial Network (MDAN) for TSC tasks. Specifically, we adopt two feature extractors for the time-domain and frequency-domain feature representations, and employ-three classifiers to perform TSC of source data for training the two feature extractors; Then, we fuse the time-domain and frequency-domain feature representations of source and target data, respectively, input them into the unified domain discriminator for unsupervised multimodal domain adversarial learning, and combine the proposed Time-Frequency-domain Joint Maximum Mean Discrepancy (TF-JMMD) to accurately align the source-target distributions; Finally, we select CNN or ResNet18 as the feature extractors to carry out comprehensive experiments, and the results demonstrate the SOTA performance of MDAN.

1. Introduction

Time series data reflect the trends of one or more variables over time, and exist in various fields [1,2]. Time Series Classification (TSC) is an important application of time series analysis and processing in data mining [3] and machine learning [4,5]. It is widely used in ECG (ElectroCardioGraph) Diagnosis [6], Mechanical Fault diagnosis [7], Gesture Recognition [8], and other scenarios. Deep learning (DL) has strong abilities for fitting and representation. Various deep neural networks, such as One-dimensional Convolutional Neural Network (CNN) [9], Long Short Term Memory (LSTM) [10], and Residual Network (ResNet) [11], have been widely used in TSC tasks.

There are mainly-two problems that make the DL models not work well for TSC tasks under some circumstances. First, many methods need large amounts of labeled data that are difficult to obtain in some real data scenarios [12,13]. Second, the training data may differ from the actual data distribution, namely Domain Shift [14], which may make it difficult for the models trained on specific datasets to be well transferred to a new scenario with different data distribution. Therefore, it

☆ Peer review under responsibility of For special issue article please include a footnote that this paper belongs to the special issue "special issue name" edited by "editor name".

* Corresponding author.
E-mail address: xiliang@hrbust.edu.cn (L. Xi).

https://doi.org/10.1016/j.ins.2022.12.069
0020-0255/

is necessary to explore an unsupervised TSC method that does not require the labels of the target data. Undoubtedly, as a manner of transfer learning, Unsupervised Domain Adaptation (UDA) is one of the ideal strategies [15].

Domain Adaptation is mainly carried out by narrowing the source-target discrepancy or adversarial training. The UDA-based methods use the labeled source data and the unlabeled target data to joint-train to reduce the Domain Shift in data distribution and perform the TSC tasks of target data [16–18]. Among the recent works on UDA, Long et al. [19,20] aim at minimizing the source-target discrepancy based on Maximum Mean Discrepancy (MMD) and Joint MMD (JMMD). The Domain-Adversarial Neural Network (DANN) [21] and Conditional Domain Adversarial Network (CDAN) [22] align the source-target distributions in an adversarial manner.

In the time series data scenarios, many methods take into account multiple modal information simultaneously. Huang et al. [23] realize an unsupervised joint learning method of feature distribution in time-domain and frequency-domain manners for anomaly detection tasks and achieves ideal performance. Therefore, we can consider multimodal information to effectively improve the feature extraction ability and align the source-target distributions. Moreover, adequate source-target discrepancy measurement is the crucial module. As a good representative, JMMD, can help the Joint Adaptation Network (JAN) [20] to obtain satisfactory source-target distribution alignment effects. However, JMMD and other discrepancy measurement methods are only suitable for single feature extraction network, while the multimodal-based UDA, proposed in this paper, has time-domain and frequency-domain feature extraction networks. Therefore, we need to improve JMMD to accommodate the multimodal data, and combine the adversarial manner to perform the UDA process for TSC tasks.

Based on the above analysis, We propose an Unsupervised **M**ultimodal **D**omain **A**dversarial **N**etwork (MDAN) for Time Series Classification: Firstly, we generate the frequency-domain data from the time-domain data in source data and target data by multiscale one-dimensional Discrete Wavelet Transform (DWT), and pre-train the corresponding time-domain and frequency-domain feature extractors with source data; Then, we design the Time-Frequency-domain Joint Maximum Mean Discrepancy (TF-JMMD) and perform the unsupervised multimodal domain adversarial learning to continue training the two feature extractors and the time–frequency-fused classifier in source data and target data; Finally, we use the trained feature extractors and time–frequency-fused classifier for the TSC tasks.

The motivation of MDAN is shown in Fig. 1. The main contributions of this paper can be summarized as follows:

1. We propose an unsupervised multimodal domain adversarial network to obtain comprehensive feature representations of time-domain and frequency-domain for TSC domain adaptation.
2. We design and add the Time-Frequency-domain Joint Maximum Mean Discrepancy (TF-JMMD) evaluation to accurately align the source-target distributions in terms of both time-domain and frequency-domain.
3. We perform different experiments on four datasets with different TSC tasks. Experimental results highlight that MDAN outperforms the baselines.

2. Related work

2.1. Unsupervised domain adaption

UDA models have shown excellent performance in many application scenarios. Long et al. [19] propose a Deep Adaptation Network (DAN) to reduce the discrepancy using an optimal multi-kernel selection method. Sun and Saenko [24] propose a Deep CORrelation ALignment (D-CORAL) method using the CORAL loss instead of MMD loss as the discrepancy measurement. Long et al. [20] present the JAN to align the source-target distributions based on JMMD. Ganin et al. [21] propose the DANN for domain adversarial adaptation by introducing a gradient reverse layer. Later, Long et al. [22] propose an improved DANN,

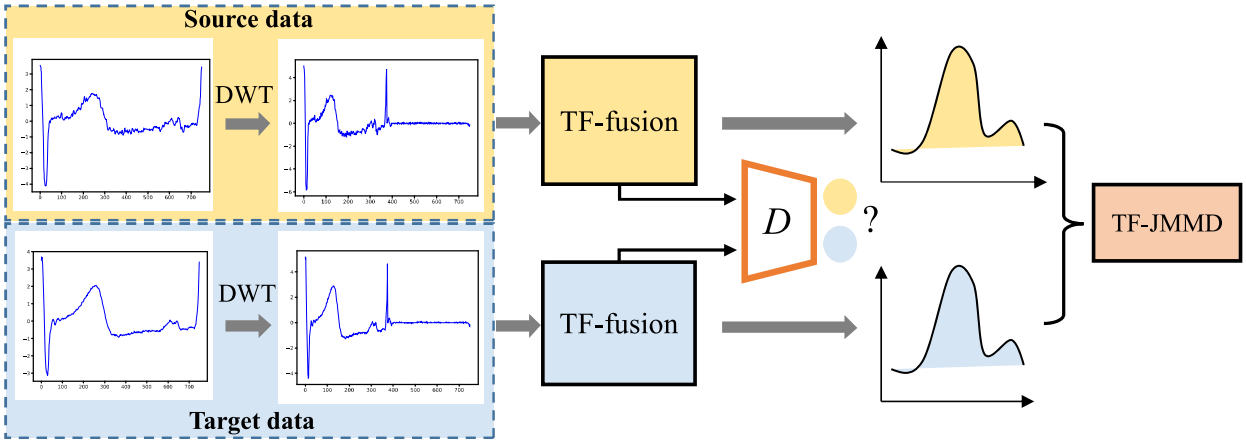

Fig. 1. The motivation of MDAN (*D* represents the domain discriminator).

CDAN, using multi-linear conditioning to enhance the discriminability, and using entropy conditioning to control the prediction uncertainty.

In the TSC application scenarios, there are also some successful UDA approaches. Guo et al. [25] present a Deep Convolutional Transfer Learning Network (DCTLN) combining MMD loss with adversarial training. Li and Zhang [26] propose a deep learning-based partial domain adaptation method by minimizing source-target MMD and unsupervised prediction consistency. Qin et al. [27] construct a Parameter Sharing Adversarial Domain Adaptation Network (PSADAN) using a shared classifier for fault classifier and domain classifier, and adding the CORAL loss for adversarial training. Zhao et al. [28] integrate the unsupervised deep transfer learning methods into a unified testing framework.

The above UDA-based TSC methods either use time series (time-domain) data as input or transform the time-domain data to frequency-domain data as input. The single-modal methods abandon the features in the other modalities and cannot fully consider the time series features in multimodal spaces. Therefore, our method considers the multimodal features to improve the transfer learning ability.

2.2. Domain-Adversarial neural Network (DANN), MMD and JMMD

Our approach borrows the ideas from DANN [21] and JMMD [20]. Let's brief them.

DANN: It is a domain adversarial learning method, to solve the problem of $P(\boldsymbol{X}_s)\neq Q(\boldsymbol{X}_t)$, where $P(\boldsymbol{X}_s)$ and $Q(\boldsymbol{X}_t)$ present the probability distributions of source data and target data, respectively. DANN includes a feature extractor, a label predictor, and a domain discriminator. During training, DANN needs to minimize the prediction loss to train the feature extractor and label predictor to predict as many true labels of source data as possible. DANN also needs to maximize the domain adversarial loss by adding a Gradient Reversal Layer (GRL), to make the domain discriminator difficult to distinguish the source-target discrepancy [28].

As stated in [22], DANN cannot capture multimodal features, and it is hard to control the domain discriminator. Moreover, DANN only adopts the adversarial strategy to align the source-target distributions, which is unstable and insufficient to obtain better transfer ability.

MMD and JMMD: Given the source and target data, $\mathscr{D}_s = \{\boldsymbol{x}_1^s, \cdots, \boldsymbol{x}_{n_s}^s\}$ and $\mathscr{D}_t = \left\{\boldsymbol{x}_1^t, ..., \boldsymbol{x}_{n_t}^t\right\}$, from the distribution, $P(\boldsymbol{X}_s)$ and $Q(\boldsymbol{X}_t)$, respectively, MMD is defined as [19]:

$$\mathscr{D}_{\mathscr{H}}(P,Q) \triangleq \left\| \mathbb{E}_{x^s}[f(\boldsymbol{x}^s)] - \mathbb{E}_{x^t}[f(\boldsymbol{x}^t)] \right\|^2_{\mathscr{H}_{\mathscr{K}}} \tag{1}$$

where $f(\cdot)$ is the feature mapping in the tensor product Hilbert space, $\mathscr{H}_{\mathscr{K}}$ represents the Reproducing Kernel Hilbert Space (RKHS), and $\mathscr{K}$ is the kernel function, generally set as a Gaussian kernel function. In practice, MMD is usually calculated using the following formula:

$$\widehat{\mathscr{D}}_{\mathscr{H}}(P,Q) = \frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s} k\left(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s\right) + \frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t} k\left(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t\right) - \frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t} k\left(\boldsymbol{x}_i^s, \boldsymbol{x}_j^t\right) \tag{2}$$

where $\widehat{\mathscr{D}}_{\mathscr{H}}(P,Q)$ is an unbiased estimator of $\mathscr{D}_{\mathscr{H}}(P,Q)$.

However, MMD is not directly defined for joint distributions of multiple domain-specific layers. Therefore, many methods employ the JMMD. JMMD is defined as:

$$D_L(P,Q) = \frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\prod_{l\in L} k^l\left(z_i^{sl}, z_j^{sl}\right) + \frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\prod_{l\in L} k\left(z_i^{tl}, z_j^{tl}\right) - \frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\prod_{l\in L} k\left(z_i^{sl}, z_j^{tl}\right) \tag{3}$$

where $\mathscr{L}$ represents the layer number of neural network. $\boldsymbol{z}_i^{s\prime}$ and $\boldsymbol{z}_i^{t\prime}$ are the feature representations of $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_i^t$ in the l-th layer, respectively.

In our method, we improve JMMD for multimodal JMMD evaluation between source data and target data and assist unsupervised multimodal domain adversarial learning.

2.3. Frequency modality of time series

There are many frequency-domain analysis and transform tools, such as Fourier Transform (FT), Short-Term Fourier Transform (STFT), Wavelet Transform (WT), etc. Li and Zhang [26], use Fast Fourier Transform (FFT) to transform time-domain data into frequency-domain data. However, FT has an inherent defect in processing non-stationary signals. It can only capture the frequency components contained in a signal. It doesn't know when the components appear, so two signals that are very different in time domain may have the same spectrum map. Shao et al. [29] adopt STFT to transform the original data into time–frequency images. STFT is well suitable for periodic and stationary signals, but it uses fixed window length [30]. Therefore, FT, FFT and STFT are not suitable for transforming non-stationary time-domain data into frequency-domain data.

WT inherits and improves the localization strategy of STFT, and uses variable window length according to frequency. WT can gradually carry out the multiscale refinement of the signal through scaling and translation, to meet the adaptive analysis

requirement of the signal details [31]. WT has been widely used in image processing [32,33], signal detection [34,35], and many nonlinear application scenarios. Since time series need discrete sampling, multiscale one-dimensional DWT is more suitable for the transform of time series data.

Given a time series sample, $\boldsymbol{x} = (x_0, x_1, \cdots, x_{M-1})^T$, M is the length. Its multiscale one-dimensional DWT is shown as follows:

$$x'_m = \frac{1}{\sqrt{M}} \sum_k W_\varphi(j_0, k)\varphi_{j_0,k}(m) + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{J} \sum_k W_\psi(j, k)\psi_{j,k}(m) \tag{4}$$

The first term is the approximate component containing the low-frequency information, and the second term is the detail component containing the high-frequency information. The noises are mainly concentrated on the detail component. $m = 0, 1, 2, ..., M-1, j = 0, 1, 2, ..., J-1$, J represents the number of transform levels, j_0 represents $j = 0$, $k = 0, 1, 2, \cdots, 2^j - 1$, represents the translation parameter, $\varphi(\cdot)$ is the scaling function, $\psi(\cdot)$ is the wavelet function, $W_\varphi(j_0, k)$ and $W_\psi(j, k)$ are the approximation coefficient and detail coefficient, respectively:

$$W_\varphi(j_0, k) = \frac{1}{\sqrt{M}} \sum\nolimits_m x_m \varphi_{j_0,k}(m) \tag{5}$$

$$W_\psi(j, k) = \frac{1}{\sqrt{M}} \sum\nolimits_m x_m \psi_{j,k}(m) \text{ for} j \geq j_0 \tag{6}$$

$$\varphi_{j,k}(m) = 2^{j/2}\varphi\left(2^j m - k\right) \tag{7}$$

$$\psi_{j,k}(m) = 2^{j/2}\psi\left(2^j m - k\right) \tag{8}$$

By applying multiscale one-dimensional DWT to each generated approximate component, we can get the low-resolution approximate component and detail component. An example of this process is shown in Fig. 2, where $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ are the approximate components in each transform level, and $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$ are the detail components in each transform level.

Using this multi-resolution characteristic, WT can conduct the time–frequency analysis on non-stationary time series to improve the TSC performance of the multimodal models. Therefore, our method employs the multiscale one-dimensional DWT for transforming time-domain data to frequency-domain data.

3. Method

MDAN is mainly composed of two feature extractors ($f_t(\cdot)$ for time-domain feature extraction, and $f_f(\cdot)$ for frequency-domain feature extraction), three classifiers ($c_t(\cdot)$ for classification in time domain, $c_f(\cdot)$ for classification in frequency domain, and $c_{tf}(\cdot)$ for final TSC by fusing the time-domain and frequency-domain feature representations), one domain discriminator, $d(\cdot)$, and the TF-JMMD evaluation, $\mathscr{L}_{TF-JMMD}$. All the classifiers are weight sharing. The domain adaption process mainly contains three stages: Frequency-domain data generation, pre-training, and unsupervised multimodal domain adversarial learning. The parameters are listed in Table 1.

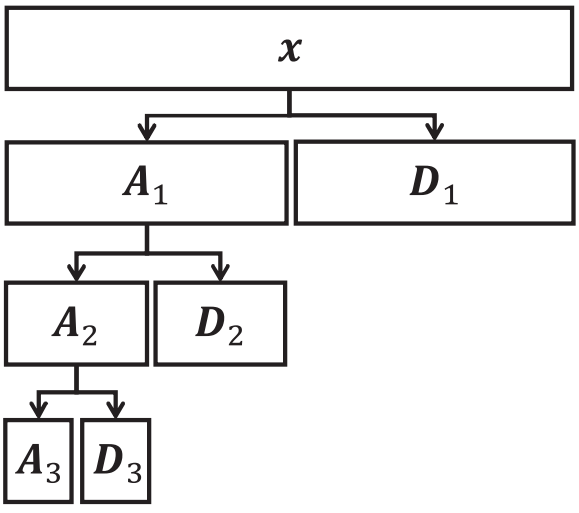

Fig. 2. Multiscale one-dimensional DWT.

Table 1
Parameter list.

Parameter	Description	Parameter	Description
$\mathscr{D}_s$	The source dataset.	$\boldsymbol{f}_t^s$	The embeddings of $\boldsymbol{x}_t^s$
$\mathscr{D}_t$	The target dataset	$\boldsymbol{f}_f^s$	The embeddings of $\boldsymbol{x}_f^s$
$\boldsymbol{x}_t^s$	A time-domain sample in source dataset	$\boldsymbol{f}_t^t$	The embeddings of $\boldsymbol{x}_t^t$
$\boldsymbol{x}_f^s$	A frequency-domain sample in source dataset	$\boldsymbol{f}_t^t$	The embeddings of $\boldsymbol{x}_f^t$
$\boldsymbol{x}_t^t$	A time-domain sample in target dataset	$\boldsymbol{f}^s$	The fusion of $\boldsymbol{f}_t^s$ and $\boldsymbol{f}_f^s$
$\boldsymbol{x}_f^t$	A frequency-domain sample in target dataset	$\boldsymbol{f}^t$	The fusion of $\boldsymbol{f}_t^t$ and $\boldsymbol{f}_f^t$

(1) Frequency-domain data generation is based on the multiscale one-dimensional DWT to transform the time-domain sample, $\boldsymbol{x}_t^s$, into the frequency-domain sample, $\boldsymbol{x}_f^s$, by formula (4).

(2) In the pre-training stage, there are two feature extractors, $f_t(\cdot)$ and $f_f(\cdot)$, and two classifiers, $c_t(\cdot)$ and $c_f(\cdot)$. The goal is to train the two feature extractors in the source data.

(3) After that, MDAN adopts all the classifiers, domain discriminator and TF-JMMD evaluation to continue training the two pre-trained feature extractors for the final TSC tasks in target data.

Definition 1. *Source dataset is denoted as*: $\mathscr{D}_{s=}\left\{\left(\boldsymbol{x}_i^s y_i^s\right)\right\}_{i=1}^{n_s}$, $\boldsymbol{x}_i^s \in \boldsymbol{X}_s$ *is the i-th sample*, $\boldsymbol{y}_i^s \in \boldsymbol{Y}_s$ *is the label of* $\boldsymbol{x}_i^s$. n_s *is the number of samples in* $\mathscr{D}_s$.

Definition 2. *Target dataset is denoted as* (*labels are not unavailable*): $\mathscr{D}_{t=}\left\{\left(\boldsymbol{x}_i^t\right)\right\}_{i=1}^{n_t}$, $\boldsymbol{x}_i^t \in \boldsymbol{X}_t$ *is the i-th sample.* n_t *is the number of samples in* $\mathscr{D}_t$.

3.1. Pre-training

The pre-training process is shown in Fig. 3. We input the time-domain samples and frequency-domain samples of source data into their respective feature extractors, $f_t(\cdot)$ and $f_f(\cdot)$, to obtain the corresponding feature representations, $\boldsymbol{f}_t^s$ and $\boldsymbol{f}_t^s$:

$$\boldsymbol{f}_t^s = f_t\left(\boldsymbol{x}_t^s; \theta_{f\_t}\right) \tag{9}$$

$$\boldsymbol{f}_f^s = f_f\left(\boldsymbol{x}_f^s; \theta_{f\_f}\right) \tag{10}$$

where θ_{f_t} and θ_{f_f} are the parameters of $f_t(\cdot)$ and $f_f(\cdot)$, respectively, which are randomly initialized and automatically adjusted during training.

Then, we employ-two classifiers, $c_t(\cdot)$ and $c_f(\cdot)$, to predict the labels of source data. Their classification loss functions are shown as follows:

$$\mathscr{L}_{c\_t} = \mathscr{L}_{c\_t}(\mathscr{D}_s, \boldsymbol{Y}_s) = \frac{1}{n_s}\sum_{i=1}^{n_s} l_{ce}\left(c_t\left(\boldsymbol{f}_t^s\right), \boldsymbol{y}_i^s; \theta_{c\_t}\right)\right) \tag{11}$$

$$\mathscr{L}_{c\_f} = \mathscr{L}_{c\_F}(\mathscr{D}_s, \boldsymbol{Y}_s) = \frac{1}{n_s}\sum_{i=1}^{n_s} l_{ce}\left(c_f\left(\boldsymbol{f}_f^s\right), \boldsymbol{y}_i^s; \theta_{c\_f}\right) \tag{12}$$

where l_{ce} is the cross entropy loss function, θ_{c_t} and θ_{c_f} are the parameters of $c_t(\cdot)$ and $c_f(\cdot)$, respectively, which are randomly initialized and automatically adjusted during training.

3.2. Multimodal domain adversarial learning

After pre-training, the feature extractors, $f_t(\cdot)$ and $f_f(\cdot)$, can effectively extract the feature representations of source data, so we perform the multimodal domain adversarial learning with source data and target data. We aim to train the $f_t(\cdot)$, $f_f(\cdot)$ and the time–frequency-fused classifier, $c_{tf}(\cdot)$, to execute the TSC tasks in the target data by unsupervised MDAN. The process is shown in Fig. 4.

Firstly, we feed the source data,$\boldsymbol{x}_t^s$ and $\boldsymbol{x}_f^s$, and target data, $\boldsymbol{x}_t^t$ and $\boldsymbol{x}_f^t$, into $f_t(\cdot)$ and $f_f(\cdot)$, respectively, to extract the corresponding feature representations, $\boldsymbol{f}_t^s$ and $\boldsymbol{f}_f^s$, $\boldsymbol{f}_t^t$ and $\boldsymbol{f}_t^t$:

Source data

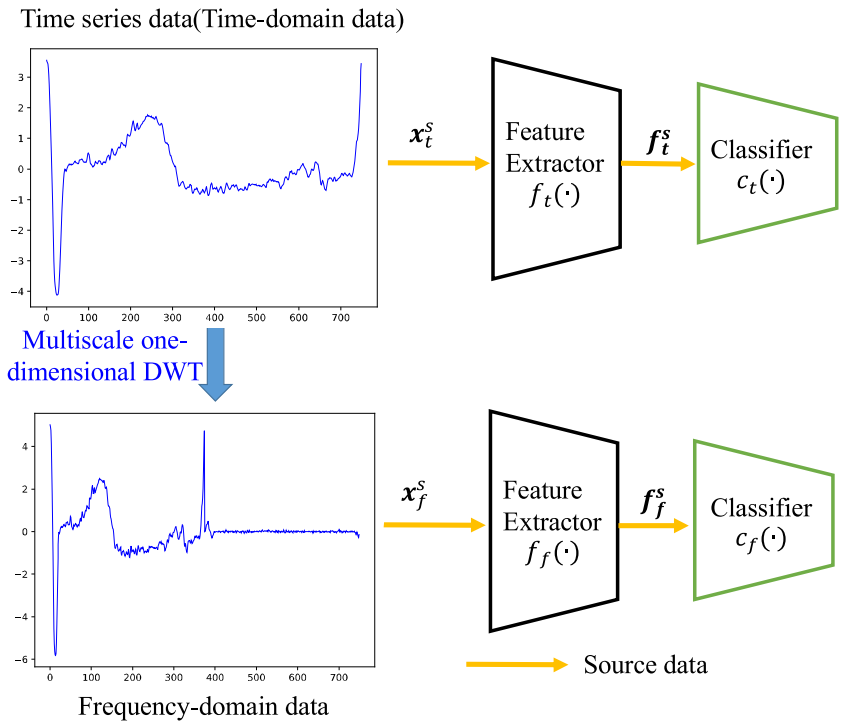

Fig. 3. Pre-training process.

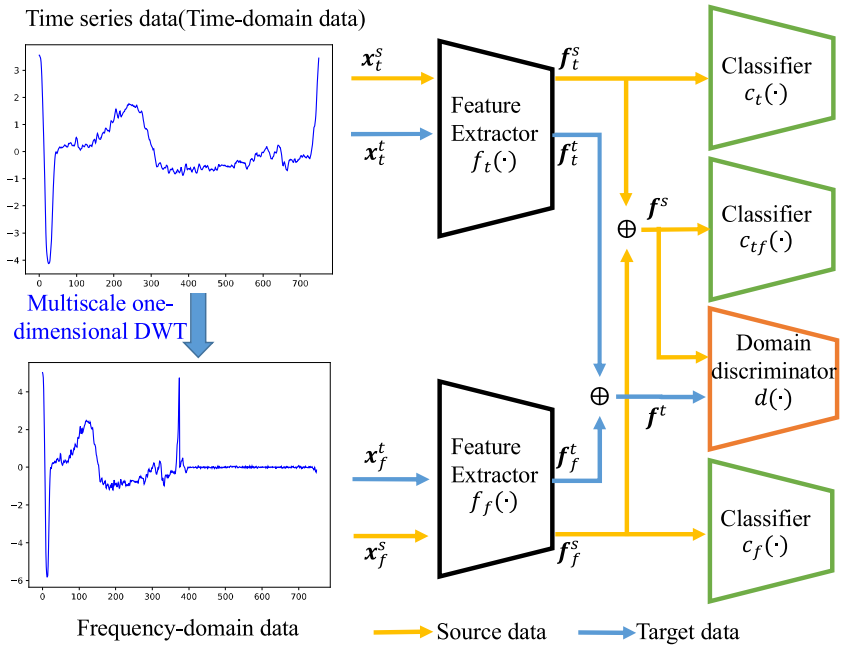

Fig. 4. Multimodal domain adversarial learning.

$$\boldsymbol{f}_t^s = f_t\left(\boldsymbol{x}_t^s; \theta_{f\_t}\right); \boldsymbol{f}_f^s = f_f\left(\boldsymbol{x}_f^s; \theta_{f\_f}\right) \tag{13}$$

Target data

$$\boldsymbol{f}_t^t = f_t(\boldsymbol{x}_t^t; \theta_{f\_t}); \boldsymbol{f}_f^t = f_f\left(\boldsymbol{x}_f^t; \theta_{f\_f}\right) \tag{14}$$

Firstly, we concat the time-domain and frequency-domain feature representations, $\boldsymbol{f}_t^s$ and $\boldsymbol{f}_f^s$, $\boldsymbol{f}_t^t$ and $\boldsymbol{f}_f^t$, to generate the fused feature representations, $\boldsymbol{f}^s$ and $\boldsymbol{f}^t$, in source data and target data, respectively:

$$f^s = f_t^s h f_f^s \tag{15}$$

$$f^t = f_t^t h f_f^t \tag{16}$$

And then, we feed $\boldsymbol{f}_t^s$ into $c_t(\cdot)$, feed $\boldsymbol{f}_f^s$ into $c_f(\cdot)$, feed $\boldsymbol{f}^s$ into $c_{tf}(\cdot)$, and feed $\boldsymbol{f}^s$ and $\boldsymbol{f}^t$ into $d(\cdot)$. $c_t(\cdot)$ and $c_f(\cdot)$ are used to assist in training $c_{tf}(\cdot)$ to obtain better accuracy of TSC in source data; $d(\cdot)$ discriminates the feature representations from the source data or the target data, and conducts adversarial training with $c_t(\cdot)$, $c_f(\cdot)$ and $c_{tf}(\cdot)$, to continue training the $f_t(\cdot)$ and $f_f(\cdot)$ to gradually extracts the domain-invariant feature representations of the source data and target data.

The loss function of $c_{tf}(\cdot)$ is shown as follows:

$$\mathscr{L}_{c\_tf} = \frac{1}{n_s}\sum\nolimits_{i=1}^{n_s} l_{ce}\left(c_{tf}\left(\boldsymbol{f}^s\right), \boldsymbol{y}_i^s; \theta_{c\_tf}\right) \tag{17}$$

where θ_{c_tf} is the parameter of $c_{tf}(\cdot)$, which is randomly initialized and automatically adjusted during training.

The total classification loss of source data is:

$$\mathscr{L}_c = \mathscr{L}_{c\_t} + \mathscr{L}_{c\_t} + \mathscr{L}_{c\_tf} \tag{18}$$

The domain adversarial loss function is shown as follows:

$$\mathscr{L}_d = \mathscr{L}_d(\mathscr{D}_s, 1) + \mathscr{L}_d(\mathscr{D}_t, 0) = -\frac{1}{n_s}\sum_{i=1}^{n_s} l_{ce}\left(d\left(\boldsymbol{f}^s\right), 1; \theta_d\right) - \frac{1}{n_t}\sum_{i=1}^{n_t} l_{ce}\left(d\left(\boldsymbol{f}^t\right), 0; \theta_d\right) \tag{19}$$

where the domain-label of source data is set as 1, and the domain-label of target data is set as 0. θ_d is the parameter of $d(\cdot)$, which is randomly initialized and automatically adjusted during training.

3.3. TF-JMMD (Time-Frequency-domain JMMD)

To further improve the stability and effectiveness of multimodal adversarial training, we design TF-JMMD based on JMMD to further accurately estimate the source-target discrepancy in time-domain, frequency-domain and the fusion of time-domain and frequency-domain (shown in Fig. 5, where $\otimes$ represents the inner product), then sum them up to obtain the total Joint Maximum Mean Discrepancy:

$$\widehat{\mathscr{D}}_{TF-JMMD}(P, Q) = \widehat{\mathscr{D}}_{c_t}(P, Q) + \widehat{\mathscr{D}}_{c_f}(P, Q) + \widehat{\mathscr{D}}_{c_{tf}}(P, Q) \tag{20}$$

Then, we can present the loss function of TF-JMMD:

$$\mathscr{L}_{TF-JMMD} = \mathscr{L}_{t-JMMD} + \mathscr{L}_{f-JMMD} + \mathscr{L}_{tf-JMMD} \tag{21}$$

where $\mathscr{L}_{t-JMMD}$, $\mathscr{L}_{f-JMMD}$ and $\mathscr{L}_{tf-JMMD}$ are the JMMD losses, respectively.

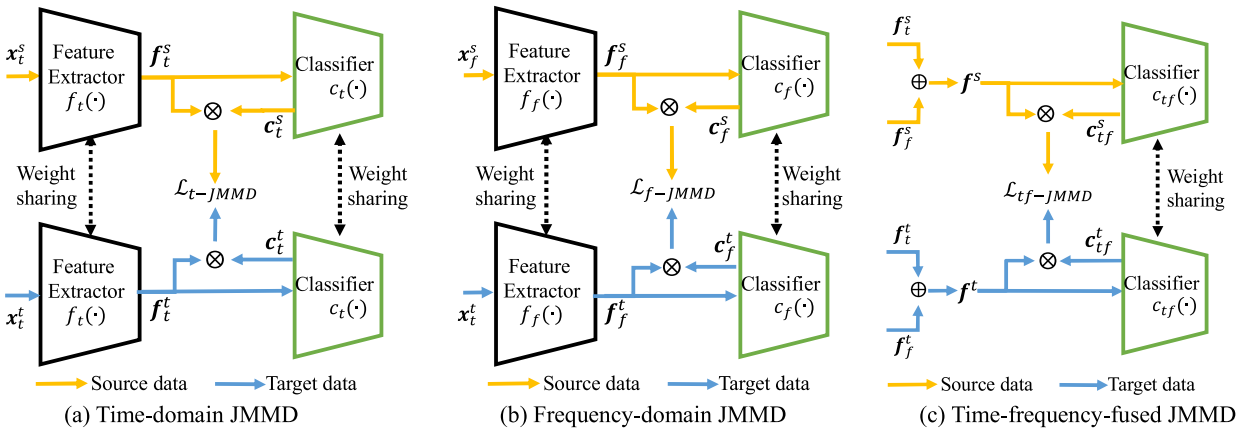

Fig. 5. Time-frequency-domain JMMD.

Finally, we can give the loss function of MDAN:

$$\mathscr{L}(\theta) = \mathscr{L}_c - \lambda\mathscr{L}_d + \lambda\mathscr{L}_{TF-JMMD} \tag{22}$$

where $\lambda \in [0,1]$, is the trade-off parameter.

Finally, the trained $f_t(\cdot)$, $f_f(\cdot)$ and $c_{tf}(\cdot)$ are used for the final TSC tasks on the target data. The process of MDAN is as follows. We use Adam Optimizer for model training.

Algorithm 1 MDAN mini-batch training process

INPUT:

Source data:$\boldsymbol{S}_t = \left\{ \left(\boldsymbol{x}_1^s, y_1^s\right), \cdots, \left(\boldsymbol{x}_{n_s}^s, y_{n_s}^s\right) \right\} \in \boldsymbol{D}_s$

Target data:$\boldsymbol{T}_t = \left\{ \left(\boldsymbol{x}_1^t\right), \cdots, \left(\boldsymbol{x}_{n_t}^t\right) \right\} \in \boldsymbol{D}_t$

BEGIN:

1:$\boldsymbol{S}_f \leftarrow \boldsymbol{S}_t$; $\boldsymbol{T}_f \leftarrow \boldsymbol{T}_t$//Frequency-domain data generation

2: Pre-train the network parameters, θ_{f_t} and θ_{f_f}, θ_{c_t} and θ_{c_f}, by $\boldsymbol{S}_t$ and $\boldsymbol{S}_f$;

// Pre-training $f_t(\cdot)$, $f_f(\cdot)$ and $c_t(\cdot)$ and $c_f(\cdot)$

3:**for** each epoch **do**

for each minibatch, $\boldsymbol{B}_{st} \subset \boldsymbol{S}_t$, $\boldsymbol{B}_{sf} \subset \boldsymbol{S}_f$, $\boldsymbol{B}_{tt} \subset \boldsymbol{T}_t$, $\boldsymbol{B}_{tf} \subset \boldsymbol{T}_f$, **do**

//Training two feature extractors, $f_t(\cdot)$ and $f_f(\cdot)$

4:$\theta_{f\_t} \leftarrow \theta_{f\_t} - h\frac{\partial L_c}{\partial \theta_{f\_t}} - \frac{\partial L_d}{\partial \theta_{f\_t}} + \frac{\partial L_{TF-JMMD}}{\partial \theta_{f\_t}} h$

5:$\theta_{f\_f} \leftarrow \theta_{f\_f} - h\frac{\partial L_c}{\partial \theta_{f\_f}} - \frac{\partial L_d}{\partial \theta_{f\_f}} + \frac{\partial L_{TF-JMMD}}{\partial \theta_{f\_f}} h$

//Training three feature extractors $c_t(\cdot)$ $c_f(\cdot)$ and $c_{tf}(\cdot)$

6:$\theta_{c\_t} \leftarrow \theta_{c\_t} + \frac{\partial L_{c_t}}{\partial \theta_{c\_t}}$

7:$\theta_{c\_f} \leftarrow \theta_{c\_f} + \frac{\partial L_{c\_f}}{\partial \theta_{c\_f}}$

8:$\theta_{c\_tf} \leftarrow \theta_{c\_tf} + \frac{\partial L_{c\_tf}}{\partial \theta_{c\_tf}}$

9:$\theta_d \leftarrow \theta_d - \frac{\partial L_d}{\partial \theta_d}$ //Domain adversarial learning

10:**end for**

11:**end for**

END.

4. Experiment

4.1. Experimental setup

The experiments were written in Python 3.7, modeled in PyTorch 1.8, and conducted on a Ubuntu-OS-based Server:
CPU: Intel Xeon Gold 5220R (24-Core, 2.2 GHz).
GPU: 2 × NVIDIA GeForce RTX 3090.
Memory: 98G RAM.

We select different types of time series datasets with different TSC tasks, shown in Table 2. For each dataset, we set the train-validation-test split as 60 %-20 %-20 %. We adopt the "Accuracy (ACC)" to evaluate the TSC performances of these methods. All the experimental results are summarized by the experiment running over five seeds.

Table 2
Statistics of these datasets.

Source	Type	Dataset	Sample	Length	Class	Task
UCR	Motion	CricketX	390	300	12	6
		CricketY	390	300	12	
		CricketZ	390	300	12	
	ECG	NIFET1	1800	750	42	2
		NIFET2	1800	750	42	
PU	Bearing	PU(0)	3255	1024	13	12
		PU(1)	3257	1024	13	
		PU(2)	3271	1024	13	
		PU(3)	3303	1024	13	
SEU	Gearbox	Bearing	3410	1024	5	2
		Gear	3410	1024	5	

(1) Cricket: It is from the UEA & UCR Time Series Classification Repository [36], and has three time series sub-datasets: CricketX, CricketY, and CricketZ. They are accelerometer data (in three dimensions) taken from actors performing cricket gestures. There are 12 classes, representing different umpire signals. We build the transfer tasks on the three sub-datasets, abbreviated as: X->Y, X->Z, Y->X, Y->Z, Z->X, Z->Y.

(2) NonInvasiveFetalECGThorax (NIFET for short): It is also from the UEA & UCR Time Series Classification Repository, and has two time series sub-datasets: NIFET1 and NIFET2. We build the transfer tasks on the two sub-datasets, abbreviated as: 1->2, 2->1.

(3) PU (Paderborn University) dataset [37,38]: It is from Paderborn University, Germany. It is a bearing time series dataset with different bearing damages. With different value combinations of load torque, radial force and rotational speed, it has four operating conditions. Therefore, there are twelve transfer tasks in total: 0->1, 0->2, 0->3, 1->0, 1->2, 1->3, 2->0, 2->1, 2->3, 3->0, 3->1, 3->2.

(4) SEU (SouthEast University) dataset [39,40]: It is a gearbox time series dataset provided by Southeast University, China. It consists of two sub-datasets: the Bearing and the Gear, which both have eight channels. In our experiment, we use the data from Channel 2. Two kinds of working conditions with "rotating speed - load configuration" set to be "20HZ−0 V" and "30HZ−2 V" are considered as two tasks: task 0 and task 1. Therefore, there are two transfer tasks in total: 0->1, 1->0.

In summary, Cricket is a relatively small dataset, and can test the transfer ability of the models in the scenarios with insufficient training samples. NIFET is a multi-class dataset, which has 42 classes, and can test the transfer ability of the models in complex TSC scenarios. PU can construct the most transfer tasks, twelve in all. In SEU, the two sub-datasets are from different components, Bearing and the Gear, and the source-target discrepancy in each transfer task is relatively large.

4.2. Comparative models

We select several currently known UDA methods for comparison with MDAN, where the former five methods are used for TSC tasks in [27], and the last method is used for TSC tasks in [28]:

(1) Basic: It uses source data to train the model, and tests with target data directly. The source data and target data share the same weight parameters.

(2) DAN [19]: It uses an optimal multi-kernel selection method to reduce the source-target discrepancy, but the source-target alignment ability is not ideal.

(3) JAN [20]: It learns a transfer network by aligning the joint distributions of multiple domain-specific layers based on JMMD, but it ignores the adversarial learning.

(4) DANN [21]: It aims to train the feature extractor and label predictor to make the domain discriminator difficult to distinguish the source-target discrepancy. The authors implemented it in both shallow and deep feed-forward architectures. However, it may not effectively align different domains of multimodal distributions.

(5) CDAN [22]: It is designed with two novel conditioning strategies: the multi-linear conditioning to improve discriminability, and the entropy conditioning to control the uncertainty of classifier predictions and guarantee the transferability. But it ignores minimizing the source-target discrepancy.

(6) PSADAN [27]: It uses a shared classifier for fault classifier and domain classifier, and adds the CORAL loss for adversarial training. But it ignores the frequency-domain information.

4.3. Parameter settings

We use CNN on PU and SEU datasets. Due to the small number of samples in Cricket and too many classes in NIFET, we select ResNet-18 [41], which has better classification performance than CNN, for transfer experiments on these two datasets.

The main parameter settings of MDAN are shown in Table 3, where the transform level, j, was tested and set in the "**Parameter sensitivity experiment**". We set the other parameters by a large number of experiments.

The parameter settings of these baselines are mainly based on their respective references, and adjusted with the optimal results obtained by many tests in each scenario of this experiment.

We use the unified code framework in [28] for the experiments. Each model is trained for 300 epochs (50 epochs for pre-training and 250 epochs for multimodal domain adversarial learning), and the training and testing processes are alternated. The batchsize is 64. We use the "step" strategy in Pytorch as the learning rate annealing method, and the initial value is 0.001 with a decay (multiplied by 0.1) in the epoch 150 and 250, respectively. For DAN, JAN, DANN, CDAN and MDAN, we use a progressive training method increasing the trade-off parameter, λ, from 0 to 1 after activating the transfer learning strategies by multiplying to $\frac{1-exp(-\gamma e)}{1+exp(-\gamma e)}$ [22], where $\gamma = 10$, $e \in [50, 250]$ and $e =$ current epoch number $- 50$. For PSADAN, we perform the experiments according to [27].

4.4. Parameter sensitivity experiment

We should determine the transform level, j, of multiscale one-dimensional DWT first. The average classification results of all transfer tasks at each value of j on each dataset are summarized as shown in Fig. 6. On Cricket and SEU datasets, the clas-

Table 3
Parameter setting of MDAN.

Module	Layer	Parameters
Feature Extractor(CNN)	Conv1d + BN + ReLU	input channel = 1, output channels = 16, kernel size = 15
	Conv1d + BN + ReLU	input channel = 16, output channels = 32, kernel size = 3
	Max Pooling	kernel size = 2, stride = 2
	Conv1d + BN + ReLU	input channel = 32, output channels = 64, kernel size = 3
	Conv1d + BN + ReLU	input channel = 64, output channels = 128, kernel size = 3
	Adaptive Max Pooling	output size = 4
	FC + ReLU	out features = 256
	Dropout	p = 0.5
Classifier	FC + Softmax	out features = number of class labels
Domain Discriminator	FC + ReLU	out features = 1024
	Dropout	p = 0.5
	FC + ReLU	out features = 1024
	Dropout	p = 0.5
	FC + Sigmoid	out features = 2

sification results are best when j is 2; On NIFET and PU datasets, the classification results are best when j is 5. Therefore, we select these optimal values for the subsequent experiments.

4.5. Ablation experiment

We design four models for the ablation experiment:

(1) Time modal DA (T-DA, i.e., DANN): The input is the original time series dataset.

(2) Frequency modal DA(F-DA): The input is the frequency-domain data obtained by multiscale one-dimensional DWT on the original time series dataset. The backbone is DANN.

(3) Multimodal DA (M−DA): The input is the collection of time-domain and frequency-domain data. The backbone is the improved DANN for multimodal data. M−DA only has one classifier, $c_{tf}(\cdot)$.

(4) Multimodal DA (M−DA) with three classifiers (M−DA$_{c3}$): It is M−DA adding two other classifiers: $c_t(\cdot)$ and $c_f(\cdot)$.

(5) Multimodal DA (M−DA) with TF-JMMD (M−DA$_{\text{TFJMMD}}$): It is M−DA adding TF-JMMD.

(6) MDAN: It is the final model of this paper, adding TF-JMMD to M−DA$_{c3}$.

The results are shown in Tables 4 and 5. We also give the average results of test time on each dataset in Table 6. MDAN outperforms the other methods on each dataset.

(1) F-DA is better than T-DA on PU and NIFET datasets, but the effects are opposite on SEU and Cricket datasets. These results suggest that there is no one modality has a universal advantage.

(2) M−DA is better than the single-modal models (T-DA and F-DA) on PU and SEU datasets; F-DA is better than M−DA on NIFET dataset; On Cricket dataset, M−DA and T-DA are generally comparable, and F-DA is the worst. These results mean that simply improving the single-modal models into the multimodal model does not guarantee that the model can fully use all the multimodal information, and the alignment method based on adversarial learning has certain defects.

(3) After improving M−DA based on TF-JMMD or adding the other classifiers, although M−DA$_{c3}$ and M−DA$_{\text{TFJMMD}}$ have achieved better results in some tasks, they are still inferior to the final model, MDAN with the three classifiers and TF-JMMD, in other tasks on each dataset. Moreover, MDAN achieves the best average results on all datasets. These results

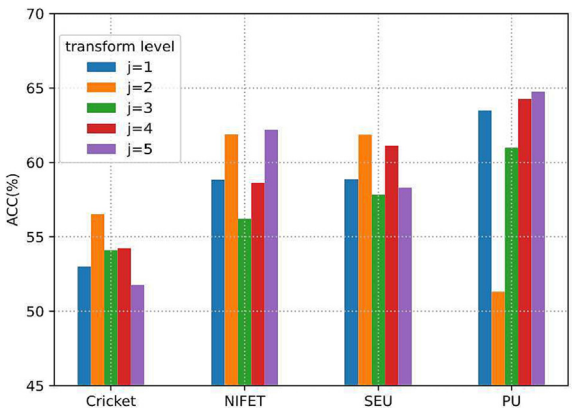

Fig. 6. TSC ACC results of MDAN with different values of j on different datasets.

Table 4
TSC ACC results (%) for the transfer tasks on Cricket, NIFET and SEU datasets.

Dataset	Cricket (X/Y/Z)							NIFET (1/2)			SEU (0/1)		
Transfer Task	X->Y	X->Z	Y->X	Y->Z	Z->X	Z->Y	AVG	1->2	2->1	AVG	0->1	1->0	AVG
T-DA	35.77 ± 1.79	**89.62 ± 0.26**	30.00 ± 5.90	33.72 ± 2.39	**89.74 ± 1.15**	37.82 ± 2.47	52.78	52.65 ± 1.79	46.37 ± 2.98	49.51	53.31 ± 1.55	61.17 ± 2.85	57.24
F-DA	32.69 ± 1.28	77.56 ± 3.01	32.82 ± 2.48	31.15 ± 2.62	75.38 ± 2.62	31.92 ± 1.48	46.92	**66.32 ± 4.25**	51.55 ± 2.10	58.94	44.93 ± 2.00	44.78 ± 2.40	44.85
M−DA	36.79 ± 2.52	88.21 ± 1.60	35.26 ± 2.87	36.41 ± 1.96	85.00 ± 1.84	34.87 ± 2.62	52.76	53.21 ± 6.41	51.98 ± 3.01	52.60	55.37 ± 3.54	64.37 ± 5.78	59.87
M−DA$_{c3}$	40.11 ± 3.38	88.59 ± 1.96	39.62 ± 2.73	**43.33 ± 1.55**	87.82 ± 1.46	39.26 ± 1.88	56.46	52.54 ± 3.35	50.23 ± 3.93	51.39	**58.30 ± 4.84**	62.73 ± 3.46	60.51
M−DA$_{TFJMMD}$	37.31 ± 4.10	85.51 ± 1.44	34.87 ± 1.32	34.74 ± 5.04	82.31 ± 0.51	32.31 ± 2.65	51.18	63.91 ± 3.20	59.39 ± 5.94	61.65	55.78 ± 1.51	0.6311 ± 6.8	59.44
MDAN	**40.26 ± 3.52**	87.69 ± 1.10	**40.64 ± 4.53**	41.15 ± 2.08	88.85 ± 1.65	**40.51 ± 2.85**	**56.52**	64.16 ± 3.79	**60.21 ± 3.33**	**62.19**	57.86 ± 2.10	**65.87 ± 3.87**	**61.86**

Table 5
TSC ACC results (%) for all the twelve transfer tasks on PU dataset.

Transfer Task	0->1	0->2	0->3	1->0	1->2	1->3	2->0	2->1	2->3	3->0	3->1	3->2	AVG
T-DA	34.97 ± 1.33	74.35 ± 3.48	51.92 ± 1.63	37.94 ± 2.86	40.67 ± 2.87	30.80 ± 1.69	74.22 ± 1.19	35.86 ± 1.93	47.08 ± 3.21	43.90 ± 2.61	31.63 ± 3.08	46.35 ± 1.23	45.81
F-DA	44.72 ± 2.91	83.73 ± 1.58	67.05 ± 0.47	48.69 ± 2.66	56.27 ± 2.85	31.83 ± 2.28	84.61 ± 1.96	45.92 ± 2.00	61.66 ± 5.83	51.49 ± 3.28	24.26 ± 1.75	54.78 ± 3.28	54.59
M−DA	52.88 ± 0.81	88.76 ± 0.70	75.73 ± 1.37	56.74 ± 3.05	57.07 ± 0.89	39.00 ± 3.32	88.63 ± 1.30	53.77 ± 1.34	69.41 ± 4.69	58.74 ± 3.51	34.26 ± 1.21	59.18 ± 2.21	61.18
M−DA_{c3}	46.44 ± 1.13	87.02 ± 0.60	65.14 ± 1.92	52.63 ± 2.23	54.90 ± 2.48	34.61 ± 1.90	85.78 ± 1.21	49.79 ± 2.96	62.48 ± 1.37	53.64 ± 3.12	33.40 ± 1.43	55.66 ± 2.68	56.79
M−DA_{TFJMMD}	56.72 ± 1.19	**90.02 ± 0.57**	76.61 ± 1.89	**61.51 ± 1.84**	**60.64 ± 1.75**	39.79 ± 2.32	88.54 ± 1.33	**57.15 ± 1.91**	**72.47 ± 3.29**	63.66 ± 4.31	34.97 ± 3.56	58.143.69	63.35
MDAN	**59.26 ± 2.35**	89.80 ± 1.28	**77.55 ± 1.03**	61.04 ± 2.86	59.57 ± 1.36	**41.45 ± 3.03**	**88.74 ± 1.47**	56.35 ± 1.35	69.68 ± 5.09	**64.95 ± 3.11**	**35.00 ± 2.24**	**59.18 ± 3.11**	**63.55**

Table 6
TSC average results of test time (unit:s) on different datasets.

Dataset	Cricket (X/Y/Z)	NIFET (1/2)	SEU (0/1)	PU (0/1/2/3)
T-DA	0.02	0.07	0.05	0.04
F-DA	0.02	0.07	0.05	0.04
M−DA	0.03	0.17	0.10	0.09
$M-DA_{c3}$	0.03	0.11	0.16	0.09
$M-DA_{TFJMMD}$	0.03	0.17	0.10	0.09
MDAN	0.03	0.17	0.18	0.09

can verify that TF-JMMD can help the adversarial learning to gain a better source-target alignment effect, serve the multimodal model to comprehensively analyze the time-domain and frequency-domain information, and effectively improve the TSC performance.

(4) From the test time results of Table 6, we also can see that the multimodal models (M−DA, $M-DA_{c3}$, $M-DA_{TFJMMD}$, MDAN) take longer to test than the single-modal models (*T*-DA and F-DA), but are acceptable. Among the multimodal models, combined with the results in Tables 4 and 5, MDAN achieves the best performance with almost the same time cost compared with the other three models ($M-DA_{c3}$, $M-DA_{TFJMMD}$, MDAN). These results show that the design of MDAN, adopting three classifiers and TF-JMMD, is meaningful.

4.6. Comparative experiment

The results are shown in Tables 7 and 8. MDAN achieves the best results on each dataset as a whole:

(1) The comparison with the domain adversarial methods (DANN, CDAN and PSADAN) can highlight MDAN can fully extract the multimodal feature representations and improve the ability of domain adversarial learning based TF-JMMD;

(2) The comparison with DAN based on MK-MMD, JAN based on JMMD, and PSADAN based on CORAL, can prove TF-JMMD of MDAN can accurately estimate the source-target discrepancy in both time-domain and frequency-domain, to align the source-target distributions and enhance the transfer ability.

(3) On individual tasks on Cricket dataset (X->Z and Z->X), MDAN does not achieve the best results, mainly because the Cricket dataset is a small dataset and highly similar among different subsets. However, MDAN has the best average and overall performance on this dataset. These results also indicate that MDAN can ensure a more stable transfer learning ability with the help of multimodal feature extraction and TF-JMMD evaluation.

(4) From the standard deviation results and the average results of all transfer tasks on the four datasets, MDAN has better stability and universality than other methods, indicating that MDAN's adversarial training based on multimodal feature extraction and TF-JMMD can converge well.

4.7. Visualization experiment

Based on the *t*-SNE tool [42], we make a visualization comparison of feature representations obtained by the models after transfer learning on PU dataset (Transfer Task: 0->3) and the SEU dataset (Transfer Task: 0->1). The results are shown in Figs. 7 and 8. We can see that MDAN gets better classification boundaries than other methods. Specifically, On PU dataset, compared with the comparative methods, MDAN can gather the data with the same class label more closely and distinguish different classes of data more obviously; On SEU dataset, in the results of comparative methods, except for the data with magenta, red and blue labels, other classes are mixed and difficult to distinguish, while MDAN makes these mixed classes more dispersed, increases their identification, and achieves a better classification effect. Moreover, the visualization results are basically consistent with the TSC results in Tables 7 and 8. These results can further verify that MDAN can comprehensively analyze multimodal information and accurately align the source-target distributions by adding TF-JMMD evaluation to improve the model's TSC performance than the baselines.

5. Conclusion

On the TSC tasks, the existing single-modal UDA methods can only learn the feature distribution in a single data space, which leads to insufficient feature extraction and low classification performance. Domain adversarial learning is an important UDA method. But it is usually unstable and does not guarantee an efficient source-target distribution alignment. Therefore, we propose an unsupervised Multimodal Domain Adversarial Network (MDAN) for TSC tasks: we adopt the time-domain and frequency-domain feature extractors and three classifiers (time-domain classifier, frequency-domain classifier, and time–frequency-fused classifier) to estimate the feature distribution of time series in the time-domain and frequency-domain spaces, and use a unified domain discriminator for adversarial training and minimizing the source-target discrepancy based on TF-JMMD. Comprehensive experimental results prove that MDAN can fully use the multimodal information for aligning the source-target distributions to improve the transfer learning ability of the model for TSC tasks. Next, we will further optimize the network parameters to enhance the effects of domain adaptation in more application scenarios.

Table 7
TSC ACC results (%) for the transfer tasks on Cricket, NIFET and SEU datasets.

Dataset	Cricket (X/Y/Z)							NIFET (1/2)			SEU (0/1)		
Transfer Task	X->Y	X->Z	Y->X	Y->Z	Z->X	Z->Y	AVG	1->2	2->1	AVG	0->1	1->0	AVG
Basic	25.90 ± 2.21	89.49 ± 1.44	30.64 ± 1.37	31.03 ± 2.71	85.38 ± 1.74	30.26 ± 1.92	48.78	28.02 ± 1.57	29.72 ± 2.02	28.87	49.88 ± 2.57	56.92 ± 3.43	53.40
DAN	33.46 ± 5.64	**90.00 ± 0.96**	30.26 ± 4.30	33.59 ± 1.19	89.49 ± 1.55	35.13 ± 3.23	51.99	54.40 ± 1.68	51.21 ± 2.59	52.80	52.58 ± 2.37	59.35 ± 3.40	55.97
JAN	33.59 ± 2.83	88.72 ± 1.55	31.79 ± 1.84	28.97 ± 4.50	89.36 ± 1.04	33.33 ± 2.69	50.96	53.26 ± 2.13	49.06 ± 1.48	51.16	51.73 ± 2.87	61.99 ± 4.57	56.86
DANN	35.77 ± 1.79	89.62 ± 0.26	30.00 ± 5.90	33.72 ± 2.39	**89.74 ± 1.15**	37.82 ± 2.47	52.78	52.65 ± 1.79	46.37 ± 2.98	49.51	53.31 ± 1.55	61.17 ± 2.85	57.24
CDAN	32.56 ± 4.54	89.74 ± 0.91	30.77 ± 1.07	33.21 ± 2.79	88.72 ± 0.65	35.13 ± 2.88	51.69	60.21 ± 2.10	53.84 ± 1.61	57.02	54.02 ± 1.34	61.35 ± 4.25	57.68
PSADAN	33.33 ± 1.67	**90.00 ± 0.96**	32.56 ± 2.85	34.87 ± 2.49	89.62 ± 1.03	37.44 ± 3.31	52.97	54.91 ± 3.00	50.76 ± 3.27	52.83	53.17 ± 1.43	63.34 ± 2.41	58.26
MDAN	**40.26 ± 1.52**	87.69 ± 1.10	**40.64 ± 2.53**	**41.15 ± 2.08**	88.85 ± 1.65	**40.51 ± 2.85**	**56.52**	**64.16 ± 2.79**	**60.21 ± 3.33**	**62.19**	**57.86 ± 2.10**	**65.87 ± 2.87**	**61.86**

Table 8
TSC ACC results (%) for all the twelve transfer tasks on PU dataset.

Transfer Task	0->1	0->2	0->3	1->0	1->2	1->3	z2->0	2->1	2->3	3->0	3->1	3->2	AVG
Basic	21.99 ± 2.48	71.91 ± 1.64	46.26 ± 4.38	29.86 ± 1.02	33.01 ± 1.88	20.76 ± 2.49	72.44 ± 1.78	20.86 ± 1.61	43.45 ± 2.38	43.38 ± 2.70	27.45 ± 1.68	41.34 ± 1.01	39.39
DAN	31.99 ± 2.84	74.14 ± 1.98	47.41 ± 1.47	36.68 ± 1.75	35.54 ± 1.30	29.17 ± 1.91	74.32 ± 1.47	32.88 ± 1.55	44.75 ± 1.83	47.56 ± 2.56	26.07 ± 0.76	48.12 ± 1.22	44.05
JAN	31.93 ± 2.57	76.82 ± 2.44	53.04 ± 1.97	37.45 ± 1.53	39.45 ± 2.65	31.13 ± 2.75	76.01 ± 1.12	37.33 ± 1.28	49.23 ± 1.25	49.83 ± 1.37	26.84 ± 1.29	49.37 ± 2.73	46.54
DANN	34.97 ± 1.33	74.35 ± 3.48	51.92 ± 1.63	37.94 ± 2.86	40.67 ± 2.87	30.80 ± 1.69	74.22 ± 1.19	35.86 ± 1.93	47.08 ± 3.21	43.90 ± 2.61	31.63 ± 3.08	46.35 ± 1.23	45.81
CDAN	33.19 ± 1.59	76.21 ± 1.10	55.16 ± 2.73	37.39 ± 2.64	40.89 ± 2.31	31.01 ± 1.52	78.19 ± 1.21	35.43 ± 3.47	49.11 ± 2.68	46.18 ± 3.53	27.15 ± 2.14	47.63 ± 1.68	46.46
PSADAN	32.64 ± 2.11	76.03 ± 2.35	54.04 ± 1.45	37.20 ± 2.87	39.48 ± 3.09	29.89 ± 2.64	74.96 ± 1.14	34.08 ± 1.80	48.05 ± 3.96	47.37 ± 2.06	27.15 ± 1.67	47.54 ± 0.90	45.70
MDAN	**59.26 ± 2.35**	**89.80 ± 1.28**	**77.55 ± 1.03**	**61.04 ± 2.86**	**59.57 ± 1.36**	**41.45 ± 3.03**	**88.74 ± 1.47**	**56.35 ± 1.35**	**69.68 ± 2.09**	**64.95 ± 2.11**	**35.00 ± 2.24**	**59.18 ± 1.11**	**63.55**

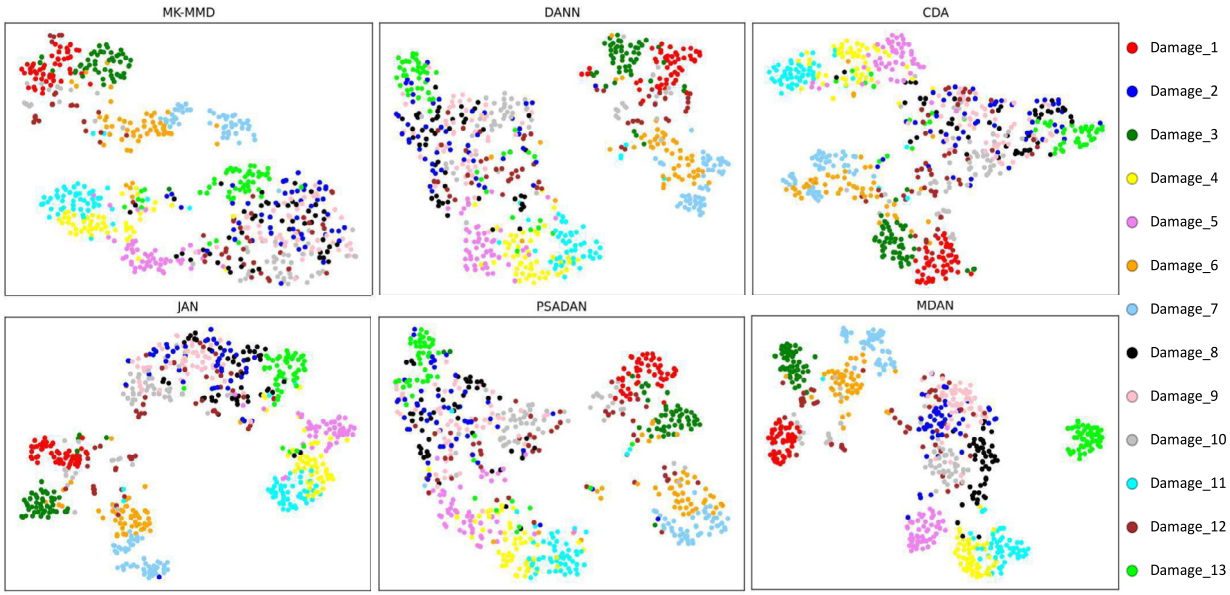

Fig. 7. Visualization results on PU dataset (Transfer Task: 0->3).

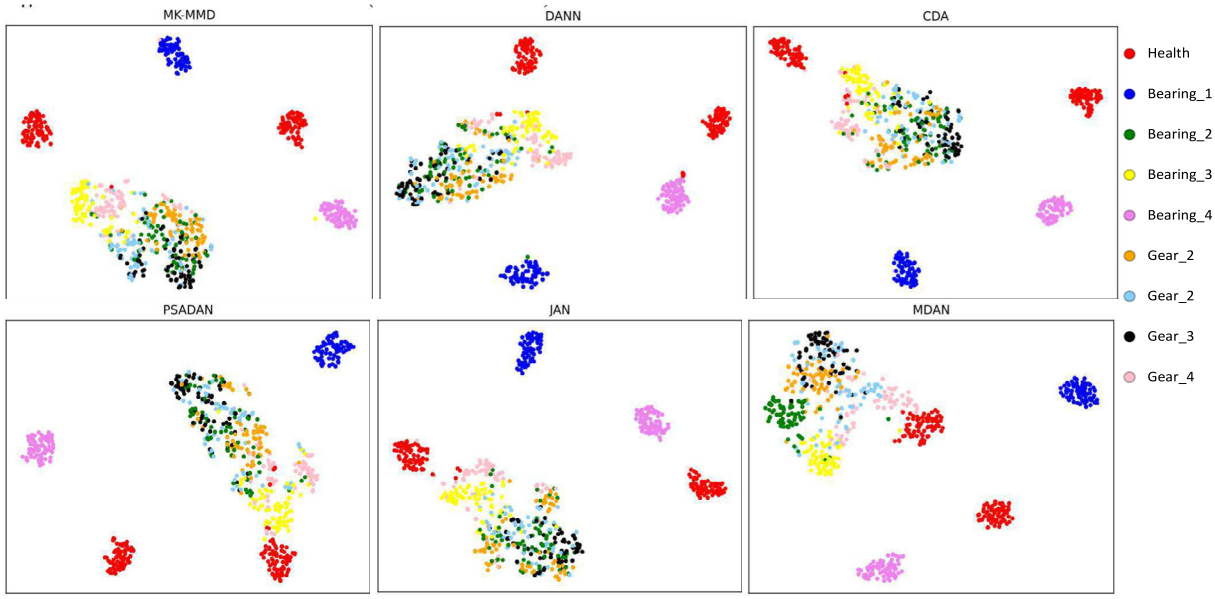

Fig. 8. Visualization results on SEU dataset (Transfer Task: 0->1).

CRediT authorship contribution statement

Liang Xi: Formal analysis, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – review & editing. **Yujia Liang:** Conceptualization, Data curation, Investigation, Methodology, Software, Writing – original draft. **Xunhua Huang:** Methodology, Software, Writing – review & editing. **Han Liu:** Methodology, Software, Writing – review & editing. **Ao Li:** Project administration, Resources, Software.

Data availability

We have shared the links to the data in the paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 62071157, and Heilongjiang Province Natural Science Foundation under Grant LH2022F034.

References

[1] Z. Xing, J. Pei, P.S. Yu, Early classification on time series, Knowl. Inf. Syst. 31 (1) (2012) 105–127, https://doi.org/10.1007/s10115-011-0400-x.
[2] Y. Tong, J. Liu, L. Yu, L. Zhang, L. Sun, W. Li, X. Ning, J. Xu, H. Qin, Q. Cai, Technology investigation on time series classification and prediction, PeerJ Comput. Sci. 8 (2022) e982.
[3] P. Esling, C. Agon, Time-series data mining, ACM Computing Surveys (CSUR) 45 (1) (2012) 1–34.
[4] D. Zhang, M. Ma, L. Xia, A comprehensive review on GANs for time-series signals, Neural Comput. Appl. 34 (5) (2022) 3551–3571.
[5] B. Ramadevi, K. Bingi, Chaotic time series forecasting approaches using machine learning techniques: a review, Symmetry 14 (5) (2022) 955, https://doi.org/10.3390/sym14050955.
[6] H. Serhal, N. Abdallah, J.-M. Marion, P. Chauvet, M. Oueidat, A. Humeau-Heurtier, Overview on prediction, detection, and classification of atrial fibrillation using wavelets and AI on ECG, Comput. Biol. Med. 142 (2022) 105168, https://doi.org/10.1016/j.compbiomed.2021.105168.
[7] F. Zhou, S. Yang, H. Fujita, D. Chen, C. Wen, Deep learning fault diagnosis method based on global optimization GAN for unbalanced data, Knowl.-Based Syst. 187 (2020), https://doi.org/10.1016/j.knosys.2019.07.008 104837.
[8] M. Wang, Z. Yan, T. Wang, P. Cai, S. Gao, Y. Zeng, C. Wan, H. Wang, L. Pan, J. Yu, S. Pan, K. He, J. Lu, X. Chen, Gesture recognition using a bioinspired learning architecture that integrates visual data with somatosensory data from stretchable sensors, Nat. Electron. 3 (9) (2020) 563–570.
[9] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Disc. 33 (4) (2019) 917–963, https://doi.org/10.1007/s10618-019-00619-1.
[10] F. Karim, S. Majumdar, H. Darabi, S. Chen, LSTM fully convolutional networks for time series classification, IEEE Access 6 (2017) 1662–1669, https://doi.org/10.1109/ACCESS.2017.2779939.
[11] Wang, Z., Yan, W., Oates, T. (2017, May). Time series classification from scratch with deep neural networks: a strong baseline. the 14th International Joint Conference on Neural Networks, Anchorage, USA, 1578-1585. 10.1109/IJCNN.2017.7966039.
[12] X. Cao, Y. Wang, B. Chen, N. Zeng, Domain-adaptive intelligence for fault diagnosis based on deep transfer learning from scientific test rigs to industrial applications, Neural Comput. Appl. 33 (9) (2021) 4483–4499, https://doi.org/10.1007/s00521-020-05275-x.
[13] H. Guan, M. Liu, Domain adaptation for medical image analysis: a survey, IEEE Trans. Biomed. Eng. 69 (3) (2021) 1173–1185, https://doi.org/10.1109/TBME.2021.3117407.
[14] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, N.D. Lawrence, (Eds.)., Dataset Shift in Machine Learning, MIT Press, 2008, pp. 1–229.
[15] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2009) 1345–1359, https://doi.org/10.1109/TKDE.2009.191.
[16] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J.W. Vaughan, A theory of learning from different domains, Mach. Learn. 79 (1) (2010) 151–175, https://doi.org/10.1007/s10994-009-5152-4.
[17] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira. Analysis of representations for domain adaptation. the 19th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 1-8. 2006.
[18] L. Bruzzone, M. Marconcini, Domain adaptation problems: a DASVM classification technique and a circular validation strategy, IEEE Trans. Pattern Anal. Mach. Intell. 32 (5) (2009) 770–787, https://doi.org/10.1109/TPAMI.2009.57.
[19] M. Long, Y. Cao, J. Wang, M. Jordan. Learning transferable features with deep adaptation networks. the 32nd International Conference on Machine Learning, Lile, France, 97-105. 2015.
[20] M. Long, H. Zhu, J. Wang, M.I. Jordan. Deep transfer learning with joint adaptation networks. the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 2208-2217. 2017.
[21] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, V. Lempitsky, Domain-adversarial training of neural networks, J. Machine Learn. Res. 17 (1) (2016) 2030–2096.
[22] M. Long, Z. Cao, J. Wang, M.I. Jordan. Conditional adversarial domain adaptation. the 32nd International Conference on Neural Information Processing Systems, Montréal Canada, 1647-1657. 10.5555/3326943.3327094. 2018.
[23] X.H. Huang, F.B. Zhang, H.Y. Fan, L. Xi, Multimodal adversarial learning based unsupervised time series anomaly detection, J. Comput. Res. Dev. 58 (8) (2021) 1655–1667, https://doi.org/10.7544/issn1000-1239.2021.20201037.
[24] B. Sun, K. Saenko. Deep CORAL: Correlation alignment for deep domain adaptation. the 14th European Conference on Computer Vision, 443-450. 10.1007/978-3-319-49409-8_35. 2016.
[25] L. Guo, Y. Lei, S. Xing, T. Yan, N. Li, Deep convolutional transfer learning network: a new method for intelligent fault diagnosis of machines with unlabeled data, IEEE Trans. Ind. Electron. 66 (9) (2018) 7316–7325, https://doi.org/10.1109/TIE.2018.2877090.
[26] X. Li, W. Zhang, Deep learning-based partial domain adaptation method on intelligent machinery fault diagnostics, IEEE Trans. Ind. Electron. 68 (5) (2020) 4351–4361, https://doi.org/10.1109/TIE.2020.2984968.
[27] Y. Qin, Q. Yao, Y. Wang, Y. Mao, Parameter sharing adversarial domain adaptation networks for fault transfer diagnosis of planetary gearboxes, Mech. Syst. Sig. Process. 160 (2021), https://doi.org/10.1016/j.ymssp.2021.107936 107936.
[28] Z. Zhao, Q. Zhang, X. Yu, C. Sun, S. Wang, R. Yan, X. Chen, Applications of unsupervised deep transfer learning to intelligent fault diagnosis: a survey and comparative study, IEEE Trans. Instrum. Meas. 70 (1–28) (2021) 3525828, https://doi.org/10.1109/TIM.2021.3116309.
[29] J. Shao, Z. Huang, J. Zhu, Transfer learning method based on adversarial domain adaption for bearing fault diagnosis, IEEE Access 8 (2020) 119421–119430, https://doi.org/10.1109/ACCESS.2020.3005243.
[30] D.D. Jayasree. Classification of power quality disturbance signals using FFT, STFT, wavelet transforms and neural networks-a comparative analysis. the 2007 International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, Tamil Nadu, India, 335-340. 10.1109/ICCIMA.2007.122. 2007.
[31] R.S. Pathak, The Wavelet Transform Vol. 4 (2009) 1–169.
[32] M.A. Gungor, K. Gencol, Developing a compression procedure based on the wavelet denoising and JPEG2000 compression, Optik 218 (2020), https://doi.org/10.1016/j.ijleo.2020.164933 164933.
[33] M.I. Anju, J. Mohan, Deep image compression with lifting scheme: wavelet transform domain based on high-frequency subband prediction, Int. J. Intell. Syst. 37 (3) (2022) 2163–2187, https://doi.org/10.1002/int.22769.
[34] W. Su, Y. Zhu, F. Liu, K. Hu, Outliers and change-points detection algorithm for time series, J. Comput. Res. Dev. 51 (4) (2014) 781–788.
[35] R. Gao, L. Du, O. Duru, K.F. Yuen, Time series forecasting based on echo state network and empirical wavelet transformation, Appl. Soft Comput. 102 (2021), https://doi.org/10.1016/j.asoc.2021.107111 107111.

[36] H.A. Dau, A. Bagnall, K. Kamgar, C.-C. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/CAA J. Autom. Sin. 6 (6) (2019) 1293–1305.
[37] C. Lessmeier, J.K. Kimotho, D. Zimmer, W. Sextro. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: a benchmark data set for data-driven classification. the 3rd European Conference of the Prognostics and Health Management Society, 1-17. 2016.
[38] C. Lessmeier, et al. KAt-DataCenter, Chair of Design and Drive Technology, Paderborn University, Available: https://mb.uni-paderborn.de/kat/forschung/datacenter/bearing-datacenter/, accessed on August 2022.
[39] S. Shao, S. McAleer, R. Yan, P. Baldi, Highly accurate machine fault diagnosis using deep transfer learning, IEEE Trans. Ind. Inf. 15 (4) (2018) 2446–2455, https://doi.org/10.1109/TII.2018.2864759.
[40] S. Shao, S. McAleer, R. Yan, P. Baldi. Mechanical dataset, Available: http://mlmechanics.ics.uci.edu./, accessed on August 2022.
[41] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. the 29th IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 770-778. 10.1109/CVPR.2016.90. 2016.
[42] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.