Pattern Recognition 109 (2021) 107617

Contents lists available at ScienceDirect

Pattern Recognition

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

Implementing transfer learning across different datasets for time series forecasting

Rui Ye, Qun Dai*

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, #29, Jiang Jun road, Nanjing 211106, China

ARTICLE INFO

Article history:
Received 26 December 2019
Revised 25 July 2020
Accepted 24 August 2020
Available online 25 August 2020

Keywords:
Time series prediction
Deep learning
Transfer learning
Convolutional neural network (CNN)

ABSTRACT

Due to the extensive practical value of time series prediction, many excellent algorithms have been proposed. Most of these methods are developed assuming that massive labeled training data are available. However, this assumption might be invalid in some actual situations. To address this limitation, a transfer learning framework with deep architectures is proposed. Since convolutional neural network (CNN) owns favorable feature extraction capability and can implement parallelization more easily, we propose a deep transfer learning method resorting to the architecture of CNN, termed as DTr-CNN for short. It can effectively alleviate the available labeled data absence and leverage useful knowledge to the current prediction. Notably, in our method, transfer learning process is implemented across different datasets. For a given target domain, in real-world scenarios, relativity of truly available potential source datasets may not be obvious, which is challenging and rarely referred to in most existing time series prediction methods. Aiming at this problem, the incorporation of Dynamic Time Warping (DTW) and Jensen-Shannon (JS) divergence is adopted for the selection of the appropriate source domain. Effectiveness of the proposed method is empirically underpinned by the experiments conducted on one group of synthetic and two groups of practical datasets. Besides, an additional experiment on NN5 dataset is conducted.

1. Introduction

Time series prediction, as a major branch of dynamic data analysis, has gathered extensive interest in many research fields [1]. In this field, various forecasting methods, such as univariate and multivariate models, local and global models, have been developed [2]. In univariate forecasting methods, the future direction of a time series is detected only by studying its past values, while multivariate methods usually model the dependency structure between the time series. Global methods are applicable to predict the demand of large amount of similar time series, where model parameters are collectively assessed based on all the time series, while in local methods, parameters are individually assessed for each time series.

Most existing time series prediction methods suppose that sufficient labeled data are available. However, this assumption is invalid in some actual applications. To address this limitation, it is desirable to leverage the knowledge digesting from the labeled data of similar nature to the target prediction task. Transfer learning can be applied to this situation [3].

As an important component of deep neural networks, convolutional neural network (CNN), provides an attractive option in the classification community [4]. Profited from the specific layered structure, CNN commonly facilitates the recognition of meaningful characteristics of input data. Its translation invariant property in time-frequency domain allows for efficient extraction and processing of features. And compared to some other neural networks, parallelization can be more easily implemented with the support of convolution operations. Recently, techniques based on CNN have also been developed for sequence modeling issues. In [5], a winning CNN-based WaveNet is proposed and successfully applied to raw audio generation techniques. In this architecture, causal and dilated convolutions are employed to ensure the data ordering and extend the receptive fields. Inspired by WaveNet, in [6], authors tactfully improve the deep convolutional networks and successfully apply it to time series. Motivated by the attractive success of CNN in various fields, in this work, a novel deep transfer learning method in virtue of the architecture of CNN, denoted as DTr-CNN, is proposed. It inherits the advantages of CNN and tries to alleviate the problem of insufficient labeled data. Due to the chronological characteristic of time series, causal convolutions is incorporated in the CNN architecture.

* Corresponding author.
E-mail address: daiqun@nuaa.edu.cn (Q. Dai).

https://doi.org/10.1016/j.patcog.2020.107617
0031-3203/© 2020 Elsevier Ltd. All rights reserved.

In many practical cases, due to information missing or high cost of manual annotations, the scarcity of labeled training data may often occur. To alleviate this problem, leveraging knowledge from different datasets to help accomplish the target prediction task is worth considering. However, though several related datasets are existent, the relevance degrees between them and the target dataset usually appear ambiguous. Blindly transferring knowledge from datasets less relevant to the target one may be a drag on the prediction performance. Considering this issue, DTr-CNN firstly, from several potential datasets, selects a most suitable one as the source dataset, and then utilizes it to help accomplish the target prediction task.

In the field of transfer learning, some excellent approaches have been proposed recently. For example, in [7], a novel Convolutional Fuzzy Sentiment Classifier is proposed. It creatively applies the combination of deep convolutional networks and fuzzy logic to sentiment analysis, and successfully solves the multi-domain and multi-modal transfer challenge. Since in many actual cases, time series, such as the univariate ones, as part of our research, may be single-modal with only one time-variate sequence. Multi-modal transfer is possibly difficult to implement under these scenarios. Hence, the proposed DTr-CNN algorithm does not consider multi-modality but focuses on extracting effective knowledge, such as the CNN network parameters, from the source domain. Unlike the implementation of transfer learning in time dimension in our previous work [8], which employs long-historical data as source domain data, transfer learning process is implemented across different datasets in this work. Since blindly employing all of the potential source datasets for training may result in unsatisfactory performance, selection of a proper source dataset is of great significance. In our method, we incorporate dynamic time warping (DTW) and Jensen-Shannon (JS) divergence to quantify the similarity between each potential source dataset and the given target dataset. In this way, both inner distributions of aligned series and shapes can be considered, which offers a general estimation of distance between the target and potential source domains. Fig. 1 intuitively shows the process of our proposed method.

The main contributions of this work can be summarized as follows:

Firstly, to alleviate the data absence problem, we construct a CNN-based transfer learning framework. Considering the chronological order of time series, causal convolution is applied to CNN layers to ensure that prediction is completed without any future data.

Secondly, we attempt to perform the transfer learning process across different datasets in this work. It is challenging because, in some real-world scenarios, some of the available potential source domains might be less related to the target task. Fondly using all of them may not yield preferable results. To address this problem, we try to select a suited source dataset based on the incorporation of DTW and JS divergence.

Thirdly, instead of transferring knowledge learned from source domain by only fine-tuning, we try to embed the transferring phase to feature learning process. In this way, the recognition of meaningful patterns similar to source samples can be accomplished. Knowledge extracted from the source domain can then be applied to the target task. Besides, since CNN exhibits superior feature extraction ability and is generally used in classification scenario, the proposed method employs stacked casual convolutions as the automatic feature extractor and applies it to forecasting problems.

The rest of this paper is organized as follows. In Section 2, we briefly review some previous work about time series analysis. In Section 3, the details of the proposed method DTr-CNN are elaborated. Experiments and corresponding analyses are implemented in Section 4. Finally, we summarize our work in Section 5.

2. Related work

2.1. Deep learning models for time series prediction

Time series prediction has been extensively applied to many practical domains. Recently, deep learning based global forecasting models have greatly facilitated the development of this field. In [6], an efficient convolutional architecture is designed for conditional time series forecasting. When forecasting, utilization of dilated convolutions extends the range of past values. The exploitation of multiple convolutional filters takes the conditioning of multivariate time series into consideration, and makes the architecture time efficient. In [9], a creative approach for the time series prediction across databases is presented. In this method, a feature-based clustering technique is employed for the optimal subgroups of time series. And then, LSTM models are separately constructed based upon each group. In [10], for time series forecasting, an approach mixing Exponential smoothing (ES) with advanced LSTM neural networks is proposed. In this method, ES, by deseasonalizing, is applied to extract the major components of each series, and advanced LSTM neural networks are used to infer the series. As an attraction of this method, local and global parameters are utilized for the construction of hierarchical structure.

2.2. Transfer learning for time series analysis

Due to the appealing performance, transfer learning has been widely applied to various situations [11]. Many efforts are also spent to detect the feasibility of transfer learning on time series analysis.

In [12], to alleviate the plight of limited training data, an appealing deep learning architecture incorporating transfer learning and new loss function is designed for time series forecasting. This LSTM-based model contains reconstruction layers and prediction layers. By computing a reconstruction loss, generalized features of original time series can be captured by reconstruction layers. The following prediction layers work as the forecasting module, with the reconstructed time series as its input, and a regression loss is used here to estimate the forecasting loss. In this method, by designing a reconstruction loss, authors creatively merge the idea of transfer learning into the architecture, and provide a more meaningful attempt to the application of transfer learning to time series.

In [13], a novel transfer learning model based on deep recurrent neural networks (RNNs) is proposed for clinical time series classification. It can be roughly divided into two parts. In the first part, a pre-trained RNN model is trained on several source datasets. And the second part is a logistic regression model, whose input is the target features obtained by the pre-trained model.

Though both the two approaches mentioned above creatively employ transfer learning for time series analysis, there are some significant differences between them and our method.

Firstly, the transferring process of the two methods are different from that of ours. Both in [12] and [13], the source dataset is only used in the pre-training process of a base model, and it is not considered when training the prediction model. While in our method, besides used for the base model, features of source data are also utilized for the generation of the final prediction model. Specifically, a Maximum Mean Discrepancy (MMD) loss, aiming to minimize the distribution discrepancy between source and target dataset, is added in our method for the extraction of target features.

Secondly, in [12], a reconstruction loss and a regression loss are employed to separately compute the modeling loss and forecasting loss. While in our method, instead of using reconstruction loss, losses aiming to minimize the feature distance between source and target datasets, termed as distance losses, are added in our objective function.

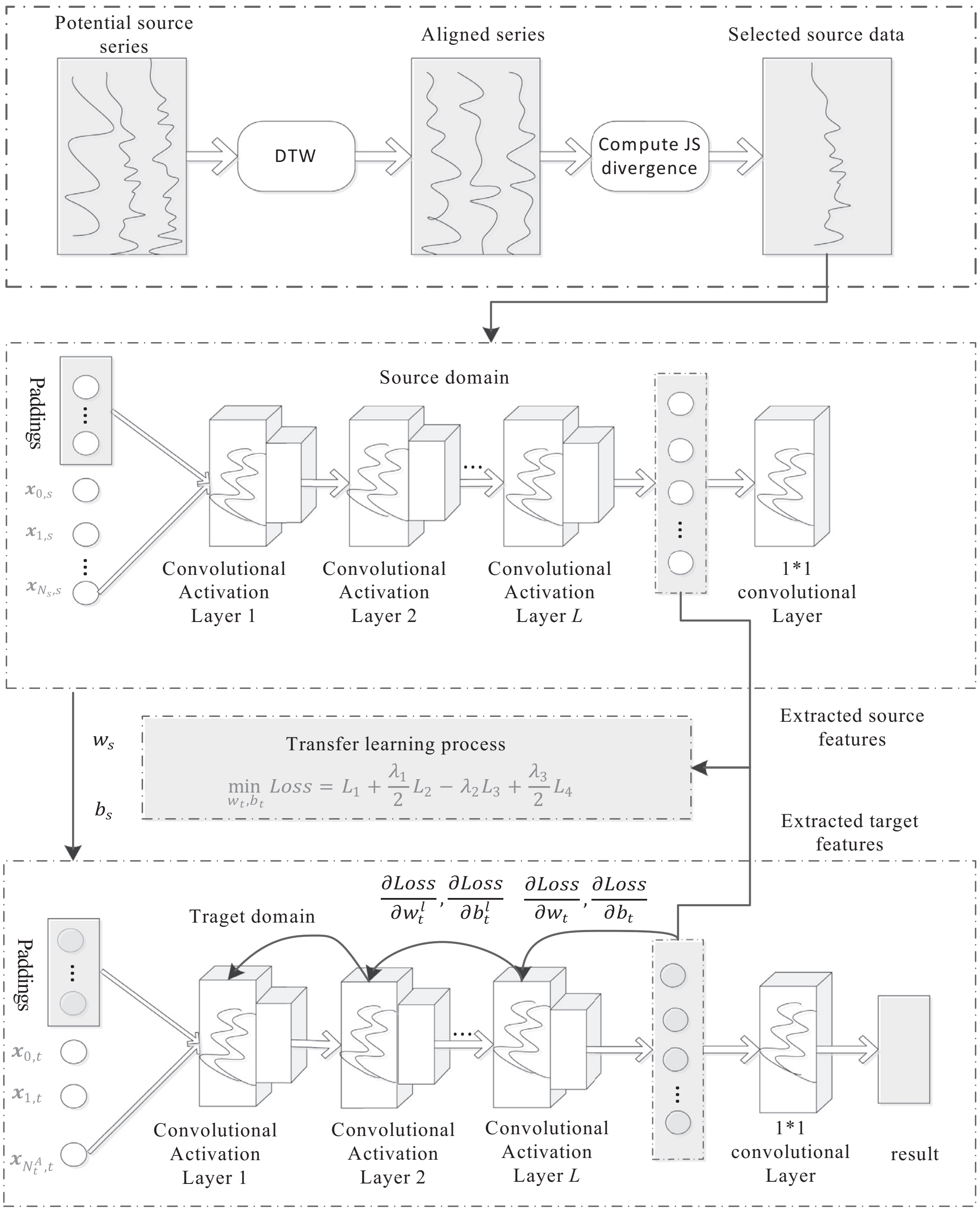
Potential source series
DTW
Aligned series
Compute JS divergence
Selected source data
Source domain
Paddings
$x_{0,s}$
$x_{1,s}$
$x_{N_s,s}$
Convolutional Activation Layer 1
Convolutional Activation Layer 2
Convolutional Activation Layer L
1*1 convolutional Layer
w_s
b_s
Transfer learning process
$\min_{w_t,b_t} Loss = L_1 + \frac{\lambda_1}{2}L_2 - \lambda_2 L_3 + \frac{\lambda_3}{2}L_4$
Extracted source features
Extracted target features
$\frac{\partial Loss}{\partial w_t^l}, \frac{\partial Loss}{\partial b_t^l}$
$\frac{\partial Loss}{\partial w_t}, \frac{\partial Loss}{\partial b_t}$
Traget domain
Paddings
$x_{0,t}$
$x_{1,t}$
$x_{N_t^A,t}$
Convolutional Activation Layer 1
Convolutional Activation Layer 2
Convolutional Activation Layer L
1*1 convolutional Layer
result

Fig. 1. The overall flow diagram of DTr-CNN.

Thirdly, in [13], after training a deep RNN base model, a logistic regression classifier is used for the final target task. While in our method, all the training and forecasting processes are completed on the CNN-based model. Besides, method in [13] is mainly applied for time series classification tasks, while method in our paper is mainly used for the forecasting tasks.

2.3. Deep learning for time series analysis

Inspired by the superior performance of deep learning in visual tasks, exploration of utilizing it for time series problems, such as classification tasks, prediction tasks and time tag tasks, etc., has been performed in recent years.

In [14], due to the temporal memory characteristic of recurrent neural networks (RNN), a trained RNN model is provided to time series classification. In [15], a competitive neuro-evolution method incorporating cooperative coevolution with recurrent neural networks is proposed for time series prediction. In [16], an efficient gradient-based method called long short-term memory (LSTM) is developed. Compared to RNN, it is effective to solve long-range time tag tasks.

3. Proposed method

In this section, a detailed introduction of the proposed method is presented. We firstly expound the way of choosing an appropriate source dataset. Secondly, a thorough explanation of our method is given.

3.1. Selection of appropriate source datasets

Reasonable source domains play a crucial role in the performance of transfer learning. An inappropriate source domain with distinct different distributions from the target task may dramatically hinder the model's performance. Hence, it is necessary to set a firm criteria for the evaluation of similarities between time series.

Here, we incorporate Dynamic Time Warping (DTW) and JS divergence to compute the similarities between different time series, for the following reasons:

Firstly, since in many cases, the data waveform usually contains some inherent information and can reflect some properties of the series, the shape similarity can, to a certain extent, help measure the distance between different series. Hence, DTW is adopted in our method. It takes shapes into consideration [17], and prefers to match series with similar shapes even they are not aligned in the time axis. That is to say, it can, to some extent, evaluate the similarity between series by performing temporal alignment.

Secondly, as a symmetric measure, JS divergence aims to measure the similarity between two probability distributions. Since DTW pays attention to the trends and shapes of series while relatively less concerns their inner distributions, JS divergence is employed to further evaluate the distribution similarity of series.

Admittedly, compared to feature-based similarity measures, our method, as a distance-based measure, has some limitations. For example, metric based on DTW may be less feasible when dealing with non-stationary data. Besides, the inherent characteristic of DTW leads to the inability of handling series with different feature dimensionality. While feature-based measures are usually more interpretable and more feasible. However, compared to distance-based measure, the feature-based ones might be more time-consuming. The same problem may be faced if apply our CNN framework itself to extract the most similar source dataset. Hence, for the time-saving reason, we use the incorporation of DTW and JS diverse, which is a distance-based measure, to compute the similarity of series. Besides, since the main goal of this paper is to discuss the transferring process of different series, the evaluation of similarity is relatively less taken consider.

Here we first briefly introduce the DTW method. Mathematical formulation of DTW is showed as Eq. (1).

$$\underset{\varepsilon_1,\varepsilon_2}{argmin} \|X_1\varepsilon_1 - X_2\varepsilon_2\|^2$$

$$\text{s.t. } \varepsilon_1 \in \{0,1\},\ \ \varepsilon_2 \in \{0,1\}, \tag{1}$$

where $X_1 \in R^{N_1\times d},\ \ X_2 \in R^{N_2\times d},\ \ \varepsilon_1 \in \{0,1\}^{d\times D},\ \ \varepsilon_2 \in \{0,1\}^{d\times D}$,$X_1$ and X_2 are two data series. ε_1 and ε_2 are the corresponding binary selection matrices.

In our case, a warping path $r = (r_1, r_2, \ldots, r_m)$ of two series can be attained by using DTW. According to the path r, sequences X_1 and X_2 can be adjusted time aligned. We denote the aligned sequences as X_1', X_2'. As a distance-based method, DTW puts emphasis on measuring the shape similarity, whereas relatively overlooks the inner distribution between time series. Hence, after alignment, we use JS divergence to further calculate the distribution similarities between two series. JS divergence is formulated as follows:

$$\mathrm{JS}(P_1(X_1')||P_2(X_2')) = \frac{1}{2}KL\left(P_1||\frac{P_1+P_2}{2}\right) + \frac{1}{2}KL\left(P_2||\frac{P_1+P_2}{2}\right),$$

$$KL(q_1||q_2) = \sum\nolimits^{q_1}(x)\log\frac{q_1(x)}{q_2(x)}, \tag{2}$$

where P, $q$represent the probability distributions of corresponding data, X_1', X_2' are time aligned sequences obtained by adjusting X_1 and X_2. Incorporation of DTW and JS divergence takes the shapes and inner distributions of aligned series into consideration, which can, to some extent, evaluate the similarity between series and guiding the selection of a source dataset for the target task.

3.2. Explanation of DTr-CNN

The inputs of time series prediction are several one-dimensional sequences and the task is to predict the value of x_{t+1} based on historical values. Due to the inherent limitations of time series, prediction value can only be influenced by its foregoing observations rather than any future value. To not violate this regulation, causal convolution is adopted in our architecture. It can be described as follows:

$$\hat{x}_{t+1} = F * x = \sum_{i=1}^{K} f_i x_{t+1-K+i}, \tag{3}$$

where $F = (f_1, f_{2,\ldots,} f_K)$ represents the filters and $\hat{x}_{t+1}$ represents the prediction value of x_{t+1}. Specifically, causal convolution can equivalently be realized by padding a set of zeros before the input time series. Then the input is fed to a CNN-liked architecture with L 1-D convolutional layers. Each layer, except the last layer, employs rectified linear unit (Relu) as activation function. The L-th layer adopts a linear activation function and is followed by a 1×1 convolutional layer, resulting in a one dimensional output by merging multiple channels into one.

To alleviate the pressure of insufficient labeled data, idea of transfer learning is borrowed to our CNN-based architecture. Many researches related to this area have been developed recently. Such as in [18], a novel deep CORAL is proposed to minimize the correlation discrepancy between source and target domains. In [19], Deep Adaptation Network (DAN), aiming to strengthen the transferability of task-specific layers, is proposed and successfully applied to domain adaptation problems. In [20], an integral probability metric is designed for minimizing the dissimilarity between domain-specific activation distributions. In [21], for the adaptation of deep classifiers across source and target domains, transferable examples are generated for the adversarial training of those classifiers. While in the proposed DTr-CNN algorithm, loss functions are designed for the alignment between source and target distributions, which is expounded in detail as follows.

Supposing that the source domain with N_s samples is denoted as $D_S = \{(\boldsymbol{x}_{1,s}, y_{1,s}), (\boldsymbol{x}_{2,s}, y_{2,s}), \ldots, (\boldsymbol{x}_{N_s,s}, y_{N_s,s})\}$, and target domain with N_t samples is denoted as $D_T = \{(\boldsymbol{x}_{1,t}, y_{1,t}), (\boldsymbol{x}_{2,t}, y_{2,t}), \ldots, (\boldsymbol{x}_{N_t,t}, y_{N_t,t})\}$. In D_T, there are N_t^A annotated samples, and the rest are unlabeled. Filters in this architecture, i.e. network weights, are trained to minimize the following objective function:

$$\min_{w_t,b_t} Loss = L_1 + \frac{\lambda_1}{2}L_2 - \lambda_2 L_3 + \frac{\lambda_3}{2}L_4, \tag{4}$$

$$L_1 = \sum_{i=1}^{N_t^A} L_t(G_t(\boldsymbol{x}_{i,t}; w_{i,t}, b_{i,t}), y_{i,t}), \tag{5}$$

$$L_2 = \left\| \frac{1}{N_t^A}\sum_{i=1}^{N_t^A} G_t^L(\boldsymbol{x}_{i,t}) - \frac{1}{N_s}\sum_{j=1}^{N_s} G_s^L(\boldsymbol{x}_{j,s}) \right\|^2, \tag{6}$$

$$L_3 = \sum_{i=1}^{N_s+N_t^A} L_h\left(H\left(G^L(\boldsymbol{x}_i), \theta\right), c_i\right), \tag{7}$$

$$L_4 = \sum_{l=1}^{L}\left(\left\|w_t^l\right\|^2 + \left\|b_t^l\right\|^2\right), \tag{8}$$

According to Eqs. (4)–(8), L_{loss} is made up of four parts. L_1 represents the training errors on the annotated target data and G_t is the feature extractor trained for target domain. It is easy to comprehend that the minimization of L_1 is aimed at reducing the prediction loss during learning stage. Since the difference of samples captured between source domain and target domain is objectively existent, to enable the feasibility of leveraging knowledge grasped from the source dataset to fulfill the target task, it is advisable to make the probability distribution of target domain, in the transformed space, as close as possible to that of the source domain. As an effective criterion for evaluating the difference between two datasets, Maximum Mean Discrepancy (MMD) can detect discrepancy between different distributions in high-performance. Here, in Eq. 6, the simplest form of MMD is utilized as the loss function L_2 to constrain the distance between features of target and source domains. Besides, to further make the target distribution as similar as possible to that of the source domain, a classifier is inserted following the feature extractor. With the goal of reducing the discrimination between source samples and target samples, we hope confusion appears when classifying the domain of extracted features. That is, parameters in G_t^L maximize the classification loss. In addition, to keep the distinguishability between source domain and target domain, the parameter θ for the classifier H aims to minimize the classification error. Note that in loss function L_{loss}, a negative factor is multiplied with L_3, aiming to reduce the dissimilarity between target and source distributions in stochastic gradient descent (SGD) process. With a view to preventing overfitting, regularization described as L_4 is added as a component of loss function. As discussed above, to align the source and target feature distributions, L_2 and L_3 are designed, respectively, from two aspects of minimizing the feature distance and maximizing the domain confusion. We adopt SGD to solve the optimization problem in Eq. (4). Detailed computation of gradients is presented as follows.

$$\begin{cases} \nabla_1 w_t^l = \dfrac{\partial L_1}{\partial w_t^l} = \sum_{i=1}^{N_t^A} \dfrac{\partial L_1}{\partial p_{i,t}^l} \cdot \dfrac{\partial p_{i,t}^l}{\partial w_{i,t}^l} = \sum_{i=1}^{N_t^A} \gamma_{i,t}^l * o_{i,t}^{l-1} \\ \nabla_1 b_t^l = \dfrac{\partial L_1}{\partial b_t^l} = \sum_{i=1}^{N_t^A} \dfrac{\partial L_1}{\partial p_{i,t}^l} \cdot \dfrac{\partial p_{i,t}^l}{\partial b_{i,t}^l} = \sum_{i=1}^{N_t^A} \gamma_{i,t}^l \end{cases} \tag{9}$$

where in Eq. (9),

$$o_{i,t}^l = G_t^l(\boldsymbol{x}_{i,t}; w_{i,t}, b_{i,t}) \tag{10}$$

$$o_{i,t}^l = \sigma\left(p_{i,t}^l\right) = \sigma\left(o_{i,t}^{l-1} * w_{i,t}^l + b_{i,t}^l\right) \tag{11}$$

$$\gamma_{i,t}^l = \frac{\partial L_1}{\partial p_{i,t}^l} = \frac{\partial L_1}{\partial p_{i,t}^{l+1}} \cdot \frac{\partial p_{i,t}^{l+1}}{\partial p_{i,t}^l} = \left(\frac{\partial p_{i,t}^{l+1}}{\partial p_{i,t}^l}\right)^T \cdot \gamma_{i,t}^{l+1}$$
$$= \gamma_{i,t}^{l+1} * ROT180\left(w_{i,t}^{l+1}\right) \odot \sigma'\left(p_{i,t}^l\right) \tag{12}$$

$$\gamma_{i,t}^L = \frac{\partial L_1}{\partial p_{i,t}^L} = \frac{\partial L_1}{\partial o_{i,t}^L} \odot \sigma'\left(p_{i,t}^L\right) \tag{13}$$

where $\odot$ is the element-wise product, $ROT180$ represents the operation of reversing the convolution kernels by 180 degree.

$$\begin{cases} \nabla_2 w_t^l = \dfrac{\partial L_2}{\partial w_t^l} = \dfrac{1}{N_t^A} \sum_{i=1}^{N_t^A} \dfrac{\partial L_2}{\partial p_{i,t}^l} \cdot \dfrac{\partial p_{i,t}^l}{\partial w_{i,t}^l} = \dfrac{1}{N_t^A} \sum_{i=1}^{N_t^A} g_{i,t}^l * o_{i,t}^{l-1} \\ \nabla_2 b_t^l = \dfrac{\partial L_2}{\partial b_t^l} = \dfrac{1}{N_t^A} \sum_{i=1}^{N_t^A} \dfrac{\partial L_2}{\partial p_{i,t}^l} \cdot \dfrac{\partial p_{i,t}^l}{\partial b_{i,t}^l} = \dfrac{1}{N_t^A} \sum_{i=1}^{N_t^A} g_{i,t}^l \end{cases} \tag{14}$$

where in Eq. (14),

$$g_{i,t}^l = \frac{\partial L_2}{\partial p_{i,t}^l} = \frac{\partial L_2}{\partial p_{i,t}^{l+1}} \cdot \frac{\partial p_{i,t}^{l+1}}{\partial p_{i,t}^l} = g_{i,t}^{l+1} * ROT180\left(w_{i,t}^{l+1}\right) \odot \sigma'\left(p_{i,t}^l\right) \tag{15}$$

$$g_{i,t}^L = \frac{\partial L_2}{\partial o_{i,t}^L} \cdot \frac{\partial o_{i,t}^L}{\partial p_{i,t}^L} = 2\left(\frac{1}{N_t^A}\sum_{i=1}^{N_t^A} o_{i,t}^L - \frac{1}{N_s}\sum_{j=1}^{N_s} o_{j,s}^L\right) \odot \sigma'(p_{i,t}^L \tag{16}$$

Since in our method, the model is first trained over the source domain, the learned knowledge is then transferred to architecture on the target domain. To effectively accomplish the transferring process, dissimilarity between extracted source features and target features should be reduced as much as possible. In L_2, we use G_s^L represents the final features extractor on source samples. Since it has little effect on the subsequent updating phase, we only take consideration on G_t^L, as described in Eq. (14), in the gradients computation of w_t^l and b_t^l. In L_3, other than parameters w_t^l and b_t^l for the trained network architecture, parameter θ is designed for the update of domain classifier.

$$\begin{cases} \nabla_3 w_t^l = \dfrac{\partial L_3}{\partial w_t^l} = \sum_{i=1}^{N_t^A} L_h' \cdot \beta_{i,t}^l * o_{i,t}^{l-1} \\ \nabla_3 b_t^l = \dfrac{\partial L_3}{\partial b_t^l} = \sum_{i=1}^{N_t^A} L_h' \cdot \beta_{i,t}^l \end{cases} \tag{17}$$

where in Eq. (7) and Eq. (17),

$$L_h\left(H\left(G^L(\boldsymbol{x}_i);\theta\right), c_i\right) = c_i \ln\left(H\left(G^L(\boldsymbol{x}_i);\theta\right)\right)$$
$$+ (1-c_i)\ln\left(1 - H\left(G^L(\boldsymbol{x}_i);\theta\right)\right) \tag{18}$$

$$H\left(G^L(\boldsymbol{x}_i);\theta\right) = \frac{1}{1+e^{-\theta^T \cdot G^L(\boldsymbol{x}_i)}} \tag{19}$$

For convenience, $H(G^L(\boldsymbol{x}_i);\ \theta)$ is abbreviated to H below.

$$\beta_{i,t}^l = \frac{\partial H}{\partial p_{i,t}^l} = \frac{\partial H}{\partial p_{i,t}^{l+1}} \cdot \frac{\partial p_{i,t}^{l+1}}{\partial p_{i,t}^l} = \beta_{i,t}^{l+1} * ROT180\left(w_{i,t}^{l+1}\right) \odot \sigma'\left(p_{i,t}^l\right) \tag{20}$$

$$\beta_{i,t}^L = \frac{\partial H}{\partial p_{i,t}^L} = \frac{\partial H}{\partial o_{i,t}^L} \cdot \frac{\partial o_{i,t}^L}{\partial p_{i,t}^L} = \frac{\partial H}{\partial o_{i,t}^L} \odot \sigma'\left(p_{i,t}^L\right) \tag{21}$$

$$\nabla\theta = \frac{\partial L_3}{\partial \theta} = \sum_{i=1}^{N_s+N_t^A} (H - c_i)\cdot G^L(x_i) \tag{22}$$

As shown in Eq. (17), since parameters in trained target network are only influenced by target features, we use the labeled target samples to compute the update of w_t^l and b_t^l. While θ is related to both source and target features, all the annotated samples, as presented in Eq. (22), are applied to the update process of θ. Constrains for w_t^l and b_t^l in L_4 are added to relieve overfitting.

$$\begin{cases} \nabla_4 w_t^l = \dfrac{\partial L_4}{\partial w_t^l} = 2w_t^l \\ \nabla_4 b_t^l = \dfrac{\partial L_4}{\partial b_t^l} = 2b_t^l \end{cases} \tag{23}$$

In conclusion, gradients of loss function for w_t^l and b_t^l can be described as follows:

$$\begin{cases} \nabla w_t^l = \nabla_1 w_t^l + \dfrac{\lambda_1}{2}\nabla_2 w_t^l - \lambda_2 \nabla_3 w_t^l + \dfrac{\lambda_3}{2}\nabla_4 w_t^l \\ \nabla b_t^l = \nabla_1 b_t^l + \dfrac{\lambda_1}{2}\nabla_2 b_t^l - \lambda_2 \nabla_3 b_t^l + \dfrac{\lambda_3}{2}\nabla_4 b_t^l \end{cases} \tag{24}$$

Algorithm 1
. The DTr-CNN algorithm.

```
Input: $D_S^{Set}$-Sets of potential source time series
    $D_T$-labeled target dataset $D_t^A = \{(\boldsymbol{x}_{1,t}, y_{1,t}), (\boldsymbol{x}_{2,t}, y_{2,t}), \ldots, (\boldsymbol{x}_{N_t^A,t}, y_{N_t^A,t})\}$
total iterative number $M$, free parameters $\lambda_1$, $\lambda_2$, $\lambda_3$, learning rate $\alpha$,
convergence error
Outputs: Parameters in trained network architecture
Phase 1:
1: Compute similarities between target dataset and time series from $D_S^{Set}$
by Eq. (2).
2: According to step 1, time series with the smallest shape and inner
distribution dissimilarity to the target dataset is selected as source
domain, denoted as $D_S = \{(\boldsymbol{x}_{1,s}, y_{1,s}), (\boldsymbol{x}_{2,s}, y_{2,s}), \ldots, (\boldsymbol{x}_{N_s,s}, y_{N_s,s})\}$.
Phase 2:
3: Train a base model on source dataset and return the extracted
features and parameters.
4: Initialize attributes of target network with parameters obtained from
step 3.
5: for m = 1,2,...,$M$
  for $l$ =1,2,...,$L$
      5.1: Do forward propagation and compute outputs $o_{i,t}^l$ of each data
point of each
layer.
      5.2: Compute auxiliary parameters according to Eqs. (13), (16) and
(21).
  end
  for $l$ = $L$,$L$-1,...,1
      5.3: Do back propagation and compute gradients according to Eqs.
(9)–(23).
  end
  for $l$=1,2,...,$L$
  5.4: update $w_t^l$ and $b_t^l$ according to Eq. (24)
  5.5: $w_t^l \leftarrow w_t^l - \alpha \nabla w_t^l$
       $b_t^l \leftarrow b_t^l - \alpha \nabla b_t^l$
  end
  If variations of $w_t^l$ and $b_t^l$ $\leq \varepsilon$, go to step 6.
end
6: Return: Weights $w_t^l$ and $b_t^l$, $l$=1,2,...,$L$
```

Formation details of DTr-CNN are introduced in Algorithm 1.

Detailed explanation of DTr-CNN is summarized in Algorithm 1. In conclusion, DTr-CNN can be approximately separated into two parts. In the first part, by using the incorporation of DTW and JS diverse, we select an appropriate time series as the source dataset. In the second part, to make it reasonable to leverage information from source domain to target task, optimization problem with the goal of minimizing both dissimilarity between domains and prediction error is addressed. Note that in this part, we first train a base network on the source data. The learned parameters of this network is reserved and passed to the training of the target network. And specifically, we can refine the construction of target network into two phases, i.e. the feature extraction phase and the forecasting phase. In the first phase, parameters are learned based on both source and target data, hence these parameters are global [10]. In the forecasting phase, that is, the final forecasting layer, parameters are specialized learned for the target time series. Hence parameters of the final layer are local and trained for individual target series. In general, method proposed in this paper can be seen as a hybrid approach having both global and local parameters.

4. Experiments

4.1. Experimental datasets

In this work, we analyze the performance of the proposed method on one group of synthetic datasets, including series from Lorenz system with different initial values, and two groups of real-world financial datasets. And the forecasting horizon of each dataset is one-step-ahead. To further test the performance of our method, additional experiment on NN5 dataset is added, whose forecasting horizon is 56.

4.1.1. Lorenz dataset

Lorenz map is a three-dimensional dynamic system with a solution of ordinary differential equations [22]. Its Mathematical expression can be described as follows:

$$\begin{aligned}\frac{dx(t)}{dt} &= \alpha_1[y(t) - x(t)]\\ \frac{dy(t)}{dt} &= x(t)[\alpha_2 - z(t)] - y(t)\\ \frac{dz(t)}{dt} &= x(t)y(t) - \alpha_3 z(t)\end{aligned} \tag{25}$$

where $(x(0), y(0), z(0))$ are initial values and α_1,α_2, α_3 are dimensionless parameters. In this paper, according to literatures, we set $\alpha_1 = 10,\ \alpha_2 = 28,\ \alpha_3 = \frac{8}{3}$ [22]. Series with different initial values are generated to compose a datasets group. Specifically, the first dataset, denoted as Lorenz_1, is generated with $(x(0) = 0.0,\ y(0) = 1.0,\ z(0) = 1.05)$. The second dataset with $(x(0) = 0.1,\ y(0) = 0.0,\ z(0) = 1.05)$ is denoted as Lorenz_2, and dataset generated with $(x(0) = 1.0,\ y(0) = 0.1,\ z(0) = 1.2)$ is denoted as Lorenz_3. Here we regard coordinate X of Lorenz_1, abbreviated as Lorenz-X1, as the target domain. Analogically, notations Lorenz-X2, Lorenz-X3 respectively represent coordinates X of Lorenz_2 and Lorenz_3, and so on. Since generation procedures of Lorenz-X2 and Lorenz-X3 are similar to that of Lorenz-X1, distributions of them may be highly alike. To improve predict difficulties, we use only Lorenz-Y2, Lorenz-Y3, Lorenz-Z2 and Lorenz-Z3 as potential source datasets. As univariate series, Lorenz-X1, Lorenz-Y2, Lorenz-Y3, Lorenz-Z2 and Lorenz-Z3 all have 2000 points in length. And the target series is divided into two parts, one for training and the other for testing.

4.1.2. Groups of real-world financial datasets

The first group contains four stock prices datasets, including data from Nov 16, 2009 to Nov 14, 2019 in Johnson Outdoors, Inc. (JOUT) dataset; data from Nov 16, 2009 to Nov 14, 2019 in Dow Jones industrial average (DJI) dataset; data from Nov 16, 2009 to Nov 14, 2019 in GlaxoSmithKline plc. (GSK) dataset; and data from Nov 16, 2009 to Nov 14, 2019 in S&P500^GSPC (SP) dataset [23]. For each dataset, we choose four attributes, including open price, highest price, lowest price and close price, in our experiment. Each of the four datasets contains 2517 series, and is sampled at a once per day frequency. When experimenting, each target dataset is divided into two parts, one for training and the other for testing.

Prediction of daily close price counts as the goal of forecasting task. Detailed mathematical expression of the task is presented below.

$$\varphi_d = f(\varphi_{d-k}, \ldots,\ \varphi_{d-1},\ \varphi_d, O_d,\ H_d, L_d), \tag{26}$$

where $\varphi_u (u = d - k, \ldots, d)$ is the u-th close price, k is the time window size, $f(\ \cdot\ )$ denotes the predictive function and O_d, H_d, L_d respectively denote the open price, the highest price and the lowest price of the d-th day.

The second group contains four stock datasets in energy service sector, including data from Feb 23, 2016 to Nov 22, 2019 in BP. plc (BP) dataset; data from Oct 24, 2016 to Feb 22, 2019 in Royal Dutch Shell plc (RDS) dataset; data from Jan 25, 2016 to Nov 22, 2019 in Total S.A. (TOT) dataset; data from Nov 23, 2016 to Nov 22, 2019 in Exxon Mobil Corporation (XOM) dataset [23]. Operation analogous to that of the first group is employed to process the second group datasets. Among these datasets, BP has 947 series. RDS has 777 series. TOT has 967 series and XOM has 755 series. Each dataset is sampled at daily granularity.

4.2. Experimental settings

To analyze the performance of the proposed DTr-CNN model, several state-of-the-art benchmarks, including Autoregressive Integrated Moving Average model (ARIMA) [24], Exponential Smoothing (ETS) [25], cross domain SVR (CDSVR), Gated Recurrent Unit (GRU) [26], deep long-short term memory (DLSTM) [27], long and short-term time series network (LSTNET) [28] and temporal convolutional network (TCN) [29], are selected as comparative methods. Among these compared approaches, LSTNET and DLSTM are global forecasting models. LSTNET can extract short-term and long-term patterns through the ingenious application of CNN and RNN. In the forecasting phase, LSTNET can be divided into neural network part and autoregressive part. And the final result is obtained by incorporating the outputs of the two parts. DLSTM involves multiple LSTM layers. Features are captured in the lower layers and passed to the higher layers. By combining the advantages of single LSTM layers, DLSTM model performs more effectively for time series forecasting task. Therein, CDSVR is inspired by the idea of [30]. To make it applicable for time series prediction issues, we replace SVM with SVR as base models. Among these methods, ETS and ARIMA are implemented by using the forecast package in [31]. Other methods, such as LSTNET [28], DLSTM [27], GRU [26], and TCN [29], are implemented by utilizing the packages provided in the corresponding references.

In addition, since DTr-CNN is capable to leverage knowledge from source domain to target domain, to make evaluation of trails more comparable, transfer framework, based on the idea brown from [11], is also constructed for the above comparative algorithms. Specifically, the models are firstly pre-trained on the source domain then re-trained on target domain through fine-tuning the whole network. For convenience, we denote these models as Tr-GRU, Tr-DLSTM, Tr-LSTNET and Tr-TCN. Besides, to evaluate the role of transfer learning in DTr-CNN, D-CNN without transfer framework is also utilized for contrast.

In our experiments, parameters of DTr-CNN are set as $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$, convergence error $= 0.0001$. Value ranges of layers L and filter size are 1 to 10, and 1 to 3, respectively. Filter number of each layer is chosen from {8, 16, 32}, and learning rate α is chosen from {0.001, 0.01, 0.1}. The Adam learning rate is set to 0.002. We tune these parameters with Randomized Search [32], which can locate the parameter values through random searches on the parameter space. In particular, we implement the Randomized Search by using the corresponding function in sklearn package [33]. In CDSVR, radial basis function $K(\xi_1, \xi_2, x) = \exp(-\xi_2 x - {\xi_1}^2),\ \ \xi_2 > 0$ is selected as the activation function. Parameters in DLSTM are determined by GA. In LSTNET, the value range of kernel size is 1 to 3, and the number of hidden nodes is chosen from {50, 100, 200} [28]. The selection of parameters is also performed by using Randomized Search. In TCN, the value range of filter size and the number of layers are respectively set to 1 to 3 and 1 to 10, and the filter number of each layer is selected from {8, 16, 32}. Analogous operations are adopted between methods and their corresponding transfer framework added methods, such as DLSTM and Tr-DLSTM, GRU and Tr-GRU, LSTNET and Tr-LSTNET, TCN and Tr-TCN.

To better contrast the performance of different algorithms, data normalization, expressed as follows, is provided in the preprocessing phase.

$$\bar{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i - \boldsymbol{x}_{min}}{\boldsymbol{x}_{max} - \boldsymbol{x}_{min}} \tag{27}$$

where $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$ respectively represents the maximum and minimum value in time series, $\boldsymbol{x}_i$ is the primitive value and $\bar{\boldsymbol{x}}_i$ is the normalized value. In our experiment, evaluation of the performances of different models is performed based on two measures, including root mean square error (RMSE) and symmetric mean absolute percentage error (sMAPE) [34]. RMSE and sMAPE are computed as follows:

Table 1
Distances between potential source datasets and target dataset.

Lorenz-Y2	Lorenz-Z2	Lorenz-Y3	Lorenz-Z3
3.1178E-2	4.8578E-1	2.4059E-2	4.7961E-1

Table 2
RMSE and sMAPE of different algorithms on Lorenz-X1.

Algorithm	RMSE	sMAPE%
CDSVR	2.5203E-2	5.7690
GRU	2.0901E-2	5.1413
Tr-GRU	2.2565E-2	5.4842
DLSTM	4.5228E-2	9.6729
Tr-DLSTM	3.5720E-2	8.6715
LSTNET	3.1779E-3	7.0805
Tr-LSTNET	2.4740E-3	5.3491E-1
TCN	3.5769E-3	7.5098E-1
Tr-TCN	1.1254E-3	2.2171E-1
D-CNN	3.5671E-3	7.1198E-1
DTr-CNN	1.0284E-3	2.6647E-1
ETS	2.7522E-2	6.7533
ARIMA	2.4051E-1	5.8934

$$RMSE = \sqrt{\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left(y_{i,pred} - y_{i,target}\right)^2} \quad i = 1, 2, \ldots, N_{test} \tag{28}$$

$$sMAPE = \frac{100\%}{N_{test}} \sum_{i=1}^{N_{test}} \frac{\left|y_{i,pred} - y_{i,target}\right|}{\left(\left|y_{i,pred} + y_{i,target}\right|\right)/2} \qquad i = 1, 2, \ldots, N_{test} \tag{29}$$

where N_{test} is the number of test samples, $y_{i,\ target}$ is the observation of the i-th sample and $y_{i,pred}$ is the prediction value.

4.3. Experimental results

4.3.1. Lorenz dataset

As described in Section 4.1, we use Lorenz-X1 as the target domain and use Lorenz-Y2, Lorenz-Z2, Lorenz-Y3, Lorenz-Z3 as potential source datasets. Here we firstly choose an appropriate source dataset. Dissimilarities between datasets are computed according to method proposed in Section 3.1. Results are shown in Table 1.

Hence, for Lorenz-X1, we employ Lorenz-Y3 as the source dataset. Table 2 records RMSE and sMAPE values of different algorithms.

Remark: In Table 2, the lowest RMSE and sMAPE values are marked in bold fonts. The remark is valid for all of the following tables.

On analysis of Table 2, on Lorenz-X1, DTr-CNN has the lowest RMSE value, and the second lowest sMAPE value, slightly higher than that of Tr-TCN. While DLSTM has relatively inferior performance. Besides, we note that in general, methods with transfer framework usually perform better on Lorenz-X1.

4.3.2. Groups of real-word financial datasets

As stated above, the first group contains four datasets, including DJI, GSK, JOUT and SP. In our experiment, each dataset is respectively regarded as the target domain and the rest are viewed as potential source domains. Table 3 presents the results of distance between datastes.

By analyzing Table 3, we can infer that the corresponding source domains of DJI, JOUT, GSK and SP are SP, DJI, JOUT and DJI. RMSE and sMAPE values are recorded as follows.

Table 3
Distances between potential source datasets and target dataset.

	DJI	GSK	JOUT	SP
DJI		2.4015E3	1.9212E3	1.3951E3
GSK	2.4013E3		2.1343E3	2.4269E3
JOUT	1.9212E3	2.1343E3		5.4707E3
SP	1.3951E3	2.4269E3	5.4707E3	

Table 4
RMSE values of different algorithms on corresponding target dataset.

RMSE	DJI(SP)	GSK(JOUT)	JOUT(DJI)	SP(DJI)
CDSVR	4.2910E-2	2.8545E-2	8.4449E-2	3.9511E-2
GRU	3.9883E-2	9.3352E-3	1.0932E-1	1.5201E-2
Tr-GRU	2.3163E-2	9.5398E-3	1.6376E-1	1.4889E-2
DLSTM	1.0800E-1	2.0389E-1	1.3897E-1	1.0308E-1
Tr-DLSTM	1.2755E-1	1.5425E-1	1.7158E-1	8.0161E-2
LSTNET	1.0116E-1	1.0291E-2	9.1347E-2	1.0570E-1
Tr-LSTNET	1.2549E-2	8.1897E-3	1.9990E-2	2.6736E-2
TCN	1.2209E-1	5.9072E-3	1.1842E-1	1.5156E-1
Tr-TCN	6.2980E-3	5.7569E-3	8.4730E-3	6.6684E-3
D-CNN	1.2715E-2	6.7927E-3	8.5870E-3	6.4192E-3
DTr-CNN	4.5511E-3	6.6162E-3	8.2962E-3	5.9534E-3
ETS	1.0224E-2	1.3256E-2	1.3825E-2	9.8131E-3
ARIMA	6.9470E-3	9,6890E-3	9.4281E-3	6.6212E-3

Table 5
sMAPE values of different algorithms on corresponding target dataset.

sMAPE%	DJI(SP)	GSK(JOUT)	JOUT(DJI)	SP(DJI)
CDSVR	5.7387	5.8196	1.8775E1	5.1055
GRU	5.7371	1.7258	2.6055E1	1.9370
Tr-GRU	3.1264	1.7997	4.1583E1	1.9252
DLSTM	1.4357E1	3.5885E1	2.7177E1	1.2862E1
Tr-DLSTM	1.7482E1	2.9768E1	2.5505E1	1.0950E1
LSTNET	1.3804E1	1.8926	1.9919E1	1.3296E1
Tr-LSTNET	1.3737	1.4970	3.2632	3.3032
TCN	1.6498E1	1.1174	2.7429E1	2.1030E1
Tr-TCN	6.0219E-1	1.1134	1.2703	6.6324E-1
D-CNN	1.6543	1.2757	1.2466	6.3043E-1
DTr-CNN	4.5626E-1	1.2358	1.2122	5.7474E-1
ETS	1.0203	2.4681	2.1662	9.5414E-1
ARIMA	6.7576E-1	1.7989	1.3563	6.3484E-1

Table 6
Distances between potential source datasets and target dataset.

	BP	RDS	TOT	XOM
BP		6.3315E2	5.9683E2	7.4166E2
RDS	6.3315E2		6.3882E2	5.5141E2
TOT	5.9683E2	6.3882E2		7.4969E2
XOM	7.4166E2	5.5141E2	7.4969E2	

Remark: Each column in this table contains RMSE values for a separate task, e.g., column DJI(SP) represents the prediction task for DJI with the selected source dataset SP. Similar notations have the same meanings in the following tables.

By analyzing Tables 4 and 5, on the four datastes, DTr-CNN gets the lowest RMSE and sMAPE values for three times. Tr-TCN performs the second best. Besides, we notice that in general, performances of methods with transfer framework are relatively better. As described in Section 4.1, in the second group, operations similar to the first group are employed. Table 6 records dissimilarities between different datasets.

By analyzing Table 6, we pair BP with TOT and pair RDS with XOM, and vice versa. RMSE and sMAPE values are listed as follows.

Due to the limited space, here we select three datasets to visually show the results of different algorithms. Since compared methods are numerous in quantity, for clarity, results of contrast methods are divided into two parts.

Remark: For clarity, the magnification image is local enlarged based on the original image. In other figures, the same markings have the same meanings.

As Fig. 2 shows, result of DTr-CNN is almost coincident with real data. D-CNN and Tr-TCN perform the next best. In boxplot, DTr-CNN, D-CNN and Tr-TCN have similar dispersals with actual data, while large deviation appears between DLSTM and real data.

As illustrated in Fig. 3, on RDS, comparatively satisfactory results are achieved by DTr-CNN, D-CNN and Tr-TCN. Conversely, DLSTM encounters difficulties on RDS with poor performance. On analysis of boxplot in Fig. 3, similar conclusion can be carried out.

As Fig. 4 shows, result curve of DTr-CNN agrees closely with the actual data. Great diversion occurs between the predictive results of Tr-DLSTM and real data. By observing the boxplot, DTr-CNN outperforms other methods with proximate dispersions to real data. Instead, large dispersion difference exists between the results of Tr-DLSTM and the actual data. Curiously, though in boxplot, compared to Tr-DLSTM, difference between DLSTM and real data is relatively small, RMSE value of DLSTM is slightly higher than that of Tr-DLSTM. This possibly because that boxplot mainly illustrates the data dispersal, while RMSE measures the deviation between predictions and real data. By analyzing figures of different algorithms' results in Fig. 4, the error between DLSTM predictions and actual values, at every time step, is relatively larger than that of Tr-DLSTM. Hence the RMSE value of DLSTM may be higher than Tr-DLSTM. But the trend of Tr-DLSTM differs enormously from the actual data, leading to large dispersion discrepancy between the two. Thus, in boxplot, DLSTM may performs better than Tr-DLSTM.

4.3.3. *Additional experiments*

In this part, experiments based on NN5 forecasting datasets and M3 datasets are added to further evaluate the proposed model. The NN5 dataset contains 111 time series of daily cash withdrawals at ATMs. Its forecast horizon is 56 days. The M3 dataset consists of 3003 time series, including yearly, monthly, quarterly and other data. Forecast horizons of these data are respectively 6, 18, 8, and 8. Herein, to evaluate our method, we aim to leverage the knowledge learned from M3 to the NN5 forecast. Before forecasting, the missing data imputation and mean normalization techniques proposed in [35] are adopted here to preprocess the data. However, since the emphasis of our paper is put on assessing the predictive and transfer performance of the proposed CNN-based transfer learning framework, for simplicity, we do not take other characteristics of time series, such as the seasonality character, into consideration. Similar to the foregoing experiments, we firstly, from M3 dataset (Yearly, Monthly, Quarterly and Other datasets), select one most similar to the NN5 dataset as the source dataset. Since both M3 and NN5 datasets contain multiple time series, we use the DTW Barycenter Averaging (DBA) [11] method to get the average series of each dataset. Then, the method introduced in Eq. 2 is adopted to calculate the distances between different datasets. Table 9 shows the dissimilarity between the M3 and NN5 datasets.

As shown in Table 9, the Yearly dataset in M3 has relatively the lowest distribution dissimilarity with NN5 dataset. Hence, here we use Yearly dataset as the source domain. Comparative methods and the corresponding settings used in this part are the same as those introduced in Section 4.2. Since forecast horizons of the two datasets are different, CDSVR may not be suitable for this scenario.

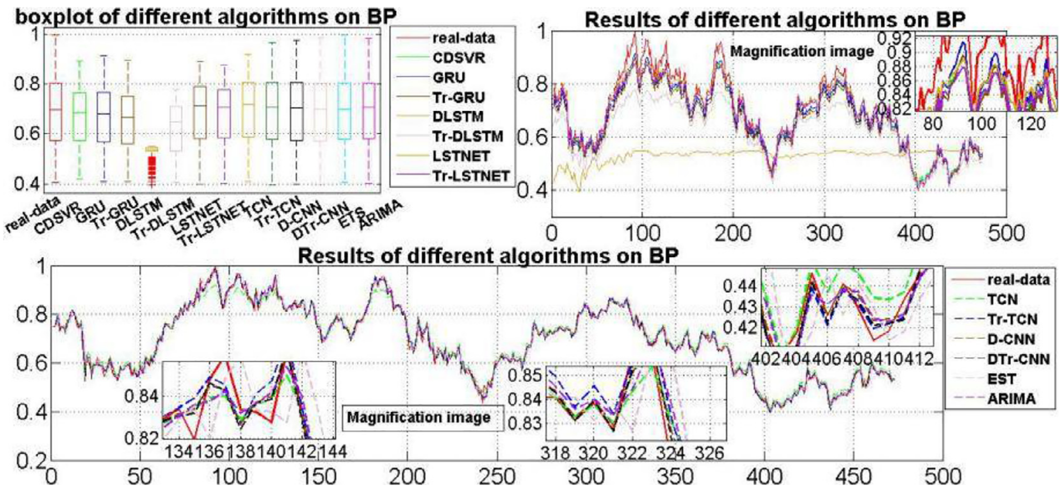

Fig. 2. Results and boxplot of different algorithms on BP.

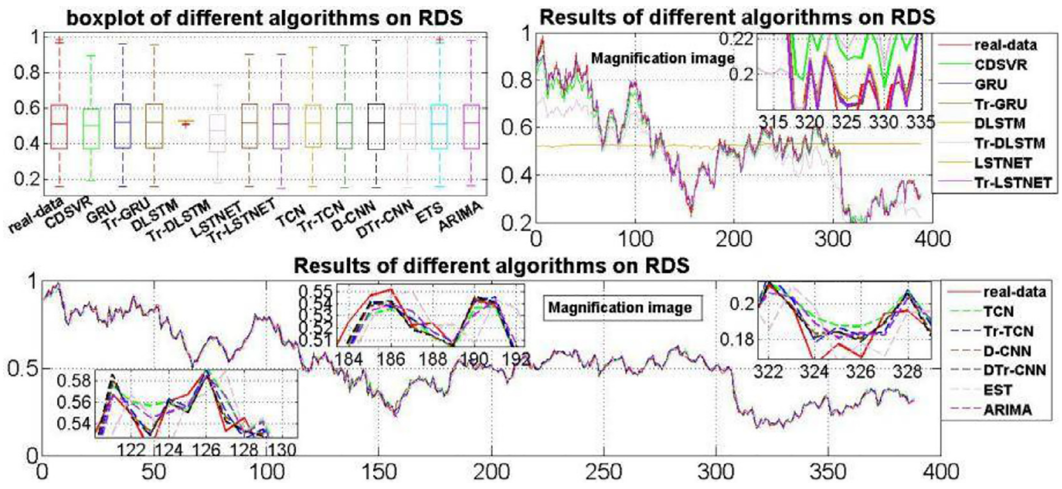

Fig. 3. Results and boxplot of different algorithms on RDS.

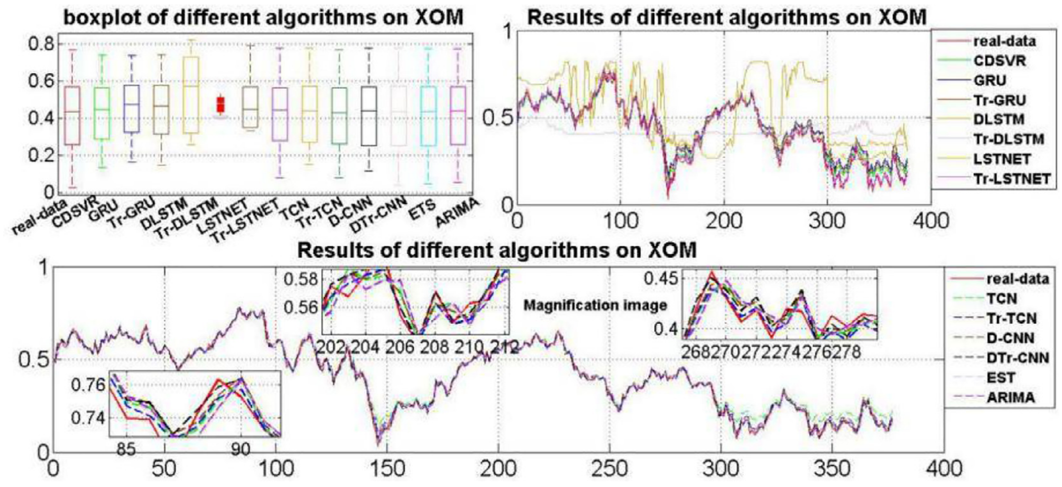

Fig. 4. Results and boxplot of different algorithms on XOM.

Table 7
RMSE values of different algorithms on corresponding target dataset.

RMSE	BP(TOT)	RDS(XOM)	TOT(BP)	XOM(RDS)
CDSVR	3.6020E-2	2.7843E-2	3.3966E-2	3.5195E-2
GRU	3.2855E-2	1.4655E-2	2.4424E-2	5.6053E-2
Tr-GRU	4.3973E-2	1.5468E-2	2.7690E-2	4.7985E-2
DLSTM	2.1487E-1	1.8917E-1	1.9012E-1	2.0915E-1
Tr-DLSTM	8.3468E-2	8.2947E-2	8.3751E-2	1.7681E-1
LSTNET	2.5973E-2	1.6270E-2	5.8073E-2	9.6273E-2
Tr-LSTNET	3.1701E-2	1.5331E-2	4.1592E-2	1.8620E-2
TCN	1.9729E-2	1.5407E-2	1.2562E-2	3.0188E-2
Tr-TCN	1.1899E-2	1.3899E-2	1.0270E-2	1.6124E-2
D-CNN	1.0634E-2	1.0440E-2	1.2916E-2	1.6196E-2
DTr-CNN	1.0026E-2	1.0172E-2	9.6076E-3	1.3203E-2
ETS	2.7357E-2	3.0691E-2	2.8688E-2	2.0547E-2
ARIMA	1.3856E-2	1.5455E-2	1.5052E-2	2.0983E-2

Table 8
sMAPE values of different algorithms on corresponding target dataset.

sMAPE%	BP(TOT)	RDS(XOM)	TOT(BP)	XOM(RDS)
CDSVR	4.5410	4.3303	4.3368	4.4625
GRU	4.2572	1.9291	3.1934	6.8718
Tr-GRU	5.8726	2.0443	3.6887	5.9633
DLSTM	3.1304E1	2.7558E1	2.6665E1	3.2949E1
Tr-DLSTM	1.1404E1	1.3936E1	1.1763E1	3.3738E1
LSTNET	2.7224	2.0976	6.8558	8.4385
Tr-LSTNET	3.5024	1.9952	4.8819	2.6097
TCN	2.1395	1.9543	1.3538	3.2486
Tr-TCN	1.2322	1.8109	1.1118	2.1742
D-CNN	1.0940	1.3733	1.5458	2.3611
DTr-CNN	1.0122	1.3347	1.0734	1.9645
ETS	2.8529	4.0122	3.2284	3.0731
ARIMA	1.3891	1.9922	1.6353	3.1259

Table 9
Distances between potential source datasets and target dataset.

	M3 (Yearly)	M3 (Monthly)	M3 (Quarterly)	M3 (Other)
NN5	5.9290E-1	6.9317E-1	6.9321E-1	6.2789E-1

Table 10
RMSE and sMAPE values of different algorithms on NN5.

Algorithm	RMSE	sMAPE %
GRU	1.9962E-1	24.5566
Tr-GRU	1.9028E-1	23.1645
DLSTM	2.0568E-1	25.4101
Tr-DLSTM	2.0380E-1	25.1125
LSTNET	1.8440E-1	22.4542
Tr-LSTNET	1.8408E-1	22.3878
TCN	2.1265E-1	25.8898
Tr-TCN	2.0955E-1	25.4835
D-CNN	1.9531E-1	23.4498
DTr-CNN	1.7391E-1	21.2435
ETS	2.2787E-1	26.8374
ARIMA	2.2415E-1	27.6214

Except for CDSVR, RMSE and sMAPE values of other algorithms are recorded as follows.

As shown in Table 10, DTr-CNN acquires the lowest RMSE and sMAPE values. LSTNET and Tr-LSTNET also perform well. Results of ETS and ARIMA compare slightly unfavorably with those reported in [35], possibly because that, except for missing data imputation and mean normalization, no other characteristics of time series are considered in the data preprocessing.

On analysis of Tables 2, 4, 5, 7–10, we can find that on ten datasets, DTr-CNN attains minimum RMSE for nine times, sMAPE for eight times, while DLSTM performs relatively poor. That may be because DLSTM suits perfectly to time series data with long time interval and long delay, while datasets in our experiment considerably varies with time and may not fully coincide with the preceding quantities. Besides, on GSK, DTr-CNN seems to be inferior to some methods. This may be caused by the large disparity between source domain and target domain. Since DTr-CNN expends much efforts on reducing dissimilarities between source and target datasets, knowledge of source domain notably influences the model's generation process. Hence, large difference could severely degrade the model's performance. By analyzing the above results, we can roughly conclude that DTr-CNN can be feasibly employed for time series forecasting.

4.3.4. Sensitive of parameters in DTr-CNN

In DTr-CNN, the layer number and the filter number are two import parameters. Since during the experiments, we discover that the other parameters, such as λ_1, λ_2, λ_3, α, have inconspicuous impacts on the performance of DTr-CNN, to save space, here we mainly discuss the sensitive of the layer number and the filter number parameters. Fig. 5 shows the average RMSE and sMAPE values, with different layer numbers and filter numbers, on all the ten experimental datasets.

From Fig. 5, we can find that DTr-CNN performs poorly when the filter number equals to 1. While the performance fluctuation is not very obvious when the filter number belongs to 8-32. Besides, the performance varies with the layer number, but modestly, when the layer number is in 3-7. Hence, we can roughly conclude that the values of the filter and layer number influence, but not significantly, the performance of DTr-CNN, when the filter and layer number separately belong to 8-32 and 3-7. That is, in this value range, the sensitivity of the two parameters is not very obvious.

5. Conclusions

Recently, innovative attempts of using deep CNN for time series forecasting and classification have been made. Such as in [11], a new method tactfully incorporating transfer learning and deep CNN is proposed for time series classification. Motivated by its attractive performance, in this paper, a novel deep transfer learning approach abbreviated as DTr-CNN, is developed for time series prediction. Compared to the method in [11], DTr-CNN extends the applied range of deep transfer CNN to forecasting issues and uses the source data both for the base model training and the target features extraction.

The major advantages of our proposed method can be summarized as follows:

Firstly, in many actual cases, insufficiency of labeled data may be an obstacle for time series prediction. Fortunately, some related datasets with sufficient labeled samples are existent. Motivated by this issue, a transfer learning framework, aiming at leveraging useful knowledge from related source datasets to the target one, is designed as DTr-CNN. This can, to some extent, recover the limitation of lacking labeled data in time series prediction.

Secondly, in the implementation, the transferring is carried out between different datasets. However, in some real-world scenarios, similarity degrees between target and available related source domains are uncontrollable. Hence, given a target dataset, we firstly pay attention to selecting a truly most similar source domain from several potential ones.

Thirdly, by minimizing the dissimilarities between source and target domains, patterns similar to source samples can be extracted from the target dataset. Then, knowledge learned from source domain can be reasonably transferred to the target task.

The proposed DTr-CNN is suitable for time series forecasting scenario, especially for the situation of lacking available training data. Besides, similarity degrees of potential source datasets and

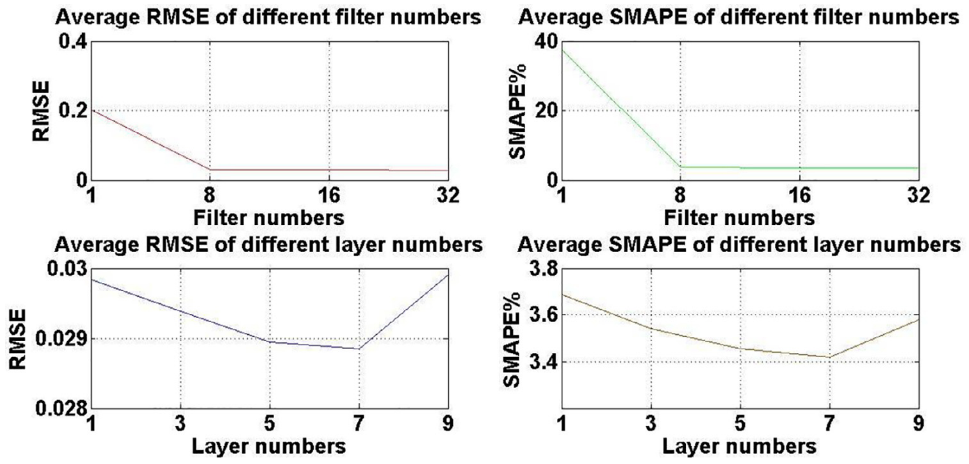

Fig. 5. Average RMSE and sMAPE of DTr-CNN with different layer and filter numbers.

the target one are considered in this algorithm, guiding to the choice of an appropriate source domain.

Also, there is some room to improve. For example, firstly, in our method, only one dataset, from several potential related datasets, is chosen as the source domain, while the others may also have some useful information for the target task. Secondly, in this paper, we mainly consider the time series with the same feature dimensionality. However, when handling sequences with varying feature dimensionality, DTW may be inappropriate. In our future work, directing at these flaws, multi-source transfer learning will be considered in our future work to yield better results. And the approach for selecting source domains should be improved when addressing more complex time series with varying feature dimensionality.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

This work is supported by the National Key R&D Program of China (Grant Nos. 2018YFC2001600, 2018YFC2001602), and the National Natural Science Foundation of China under Grant no. 61473150.

References

[1] K.W. Lau, Q.H. Wu, Local prediction of non-linear time series using support vector regression, Pattern Recognit. 41 (5) (May 2008) 1539–1547.
[2] T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, Criteria for classifying forecasting methods, Int. J. Forecast. 36 (2020) 167–177.
[3] B. Huang, T. Xu, J. Li, Z. Shen, Y. Chen, Transfer learning-based discriminative correlation filter for visual tracking, Pattern Recognit. 100 (2020) 107–157.
[4] C.L. Zhang, J. Wu, Improving CNN linear layers with power mean non-linearity, Pattern Recognit. 89 (2019) 12–21 May.
[5] A.V.D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, "WaveNet: a generative model for raw audio," arXiv:1609.03499, Sep 2016.
[6] A. Borovykh, S. Bohte, C.W. Oosterlee, Conditional time series forecasting with convolutional neural networks, Artif. Neural Netw. Mach. Learn. 10614 (2017) 729–750.
[7] I. Chaturvedi, R. Satapathy, S. Cavallari, E. Cambria, Fuzzy commonsense reasoning for multimodal sentiment analysis, Pattern Recognit. Lett. 125 (2019) 264–270.
[8] R. Ye, Q. Dai, A novel transfer learning framework for time series forecasting, Knowl.-Based Syst. 156 (2018) 74–99.
[9] K. Bandara, C. Bergmeir, S. Smyl, Forecasting across time series databases using recurrent neural networks on groups of similar series: a clustering approach, Expert Syst. Appl. 140 (2020) 112–127.
[10] S. Smyl, A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, Int. J. Forecast. 36 (2020) 75–85.
[11] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.A. Muller, Transfer learning for time series classification, in: 2018 IEEE International Conference on Big Data, 2018, pp. 1367–1376.
[12] N. Laptev, Y. Jiafan, R. Rajagopal, Reconstruction and regression loss for time-series transfer learning, SIGKDD MiLeTS'18, 2018.
[13] P. Gupta, P. Malhotra, L. Vig, and G. Shroff, "Transfer learning for clinical time series analysis using recurrent neural networks," arXiv:1807.01705, 2018.
[14] M. Hüsken, P. Stagge, Recurrent neural networks for time series classification, Neurocomputing 50 (2003) 223–235.
[15] R. Chandra, Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction, IEEE Trans. Neural Netw. Learn. Syst. 26 (2015) 3123–3136.
[16] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[17] E. Keogh, C.A. Ratanamahatana, Exact indexing of dynamic time warping, Knowl. Inf. Syst. 7 (2005) 358–386.
[18] B. Sun, K. Saenko, Deep CORAL: correlation alignment for deep domain adaptation, ECCV 2016: Computer Vision 9915 (2016) 443–450.
[19] M. Long, Y. Cao, J. Wang, M.I. Jordan, Learning transferable features with deep adaptation networks, in: International Conference on Machine Learning (ICML), 37, 2015, pp. 97–105.
[20] W. Zellinger, B.A. Moser, T. Grubinger, E. Lughofer, T. Natschläger, S. Saminger-Platz, Robust unsupervised domain adaptation for neural networks via moment alignment, Inf. Sci. 483 (2019) 174–191.
[21] M. Long, H. Liu, J. Wang, M.I. Jordan, Transferable adversarial training: a general approach to adapting deep classifiers, in: International Conference on Machine Learning (ICML), 97, 2019, pp. 4013–4022.
[22] M. Ardalanifarsa, S. Zolfaghari, Chaotic time series prediction with residual analysis method using hybrid Elman-NARX neural networks, Neurocomputing 73 (2010) 2540–2553.
[23] Y. Finance, Available: http://finance.yahoo.com/.
[24] O.S. Makinde, O.A. Fasoranbaku, Identification of optimal autoregressive integrated moving average model on temperature data, J. Mod. Appl. Stat. Methods 10 (2011) 718–729.
[25] C. Chatfield, A.B. Koehler, J.K. Ord, R.D. Snyder, A new look at models for exponential smoothing, J. Roy. Statist. Soc. Ser. D (The Statistician) 50 (2001) 147–159.
[26] K. Cho, Learning phrase representations using RNN encoder-decoder for statistical machine translation, Empir. Methods Nat Lang. Process. (2014) 1724–1734.
[27] A.S.a.M. Kotb, Time series forecasting of petroleum production using deep LSTM recurrent networks, Neurocomputing 323 (2019) 203–213.
[28] G. Lai, W.C. Chang, Y. Yang, H. Liu, Modeling long and short term temporal patterns with deep neural networks, in: ACM/SIGIR Proceedings 2018, 2018, pp. 95–104.
[29] S. Bai, J.Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv:1803.01271, 2018.
[30] W. Jiang, E. Zavesky, S.F. Chang, A. Loui, Cross-domain learning methods for high-level visual concept classification, in: IEEE International Conference on Image Processing, IEEE, 2008, pp. 161–164.
[31] forecast package (2020) https://cran.r-project.org/web/packages/forecast/forecast.pdf.
[32] S. Droste, T. Jansen, I. Wegener, Optimization with randomized search heuristics-the (A)NFL theorem, realistic scenarios, and difficult functions, Theor. Comput. Sci. 287 (2002) 131–144.
[33] Sklearn Package (2015). Available: https://pypi.org/project/sklearn/.
[34] R.J. Hyndman, A.B. Koehler, Another look at measures of forecast accuracy, Int. J. Forecast. 22 (2006) 679–688.
[35] H. Hewamalage, C. Bergmeir, K. Bandara, "Recurrent neural networks for time

series forecasting: current status and future directions," arXiv:1909.00590, 2019. https://arxiv.org/pdf/1909.00590.pdf

Rui Ye received a bachelor's Degree in College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. Then she entered Nanjing University of Aeronautics and Astronautics, Nanjing, China, in the same year as a graduate student. She completed her M.S. degree in computer science at Nanjing University of Aeronautics and Astronautics (NUAA) in March, 2019, and then she became a Ph. D candidate in NUAA. Her research interests include pattern recognition, intelligent systems and machine learning.

Qun Dai completed her M.S. degree in computer science at Nanjing University of Aeronautics and Astronautics (NUAA) in March, 2003, and then worked at the College of Information Science and Technology of NUAA as an assistant lecturer. There she received a Ph.D. degree in computer science in 2009. In 2010, she became an Associate Professor for the College of Computer Science and Technology of NUAA. Since 2015, she has become a Professor for the College of Computer Science and Technology of NUAA. Her research interests focus on neural computing, pattern recognition and machine learning.