Energy 217 (2021) 119356

Contents lists available at ScienceDirect

Energy

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

Short-term predictions of multiple wind turbine power outputs based on deep neural networks with transfer learning

Xin Liu [a], Zheming Cao [b], Zijun Zhang [a, *]

[a] School of Data Science, City University of Hong Kong, Hong Kong
[b] Taiji Computer Co. Ltd., No. 7 Rongda Road, Chaoyang District, Beijing, 100102, China

ARTICLE INFO

Article history:
Received 5 June 2020
Received in revised form
7 November 2020
Accepted 15 November 2020
Available online 18 November 2020

Keywords:
Wind power
Data-driven method
SCADA data
Short-term prediction
Neural networks

ABSTRACT

This paper proposes a novel deep and transfer learning (DETL) framework, which enables a more efficient development of data-driven wind power prediction models for a group of wind turbines. In DETL, a transfer learning scheme is developed to boost computations in modeling wind power generation processes from a data-driven perspective and derive latent features for conducting power predictions. To perform the transfer learning, a new data organization scheme, which separates a batch of wind turbine datasets into a source domain and multiple target domains, is adopted. Based on the source domain, the DETL attempts to extract homogeneous characteristics of multiple wind turbine system dynamics via developing a base Auto-encoder (AE), whose architecture is adaptively determined. Next, the DETL aims to specify heterogeneous characteristics among individual wind turbine system dynamics via learning target domains, which converts the base AE model into multiple customized AE models. Finally, the customized AE model representing system dynamics of each wind turbine is extended to conduct multi-step wind power predictions by additionally incorporating temporal features and prediction targets. Field data collected from 50 wind turbines in commercial wind farms are utilized to verify the proposed DETL. Computational experiments validate that the DETL outperforms conventional training methods on developing a batch of prediction models with a higher prediction accuracy and faster training speed.

1. Introduction

Wind energy, as one of the clean and renewable energy sources, has been considered as an important supplement to fossil energy in many countries. However, the stochastic nature of the wind causes a great uncertainty in the wind power generation. To make power grid operations more manageable, the utility of wind power is often curtailed due to its intermittency. Therefore, to boost the wind power penetration in a power grid, an accurate wind power prediction is of great significance.

A commercial wind farm typically contains dozens of wind turbines (WT) spatially located over a broad wild. Since WTs are key assets in wind power generations, modeling and analyzing dynamics of WTs are crucial for studying the wind power prediction problem. Nowadays, the supervisory control and data acquisition (SCADA) systems have been commonly deployed in wind farms, which enables an unprecedented opportunity of the WT data collection. Abundant WT data provides a solid platform for exploring WT power generation characteristics from a data-driven perspective. To generate wind power predictions, previous studies majorly developed wind power prediction models from three major aspects, physics-based approaches [1–3], classical data-driven approaches [4–9], and deep learning approaches [10–13].

In physics-based studies, numerical weather predictions (NWP) are usually first performed to predict weather conditions, such as the wind speed, temperature, humidity, etc. [1] Next, predicted weather conditions are fed into physics-based models to generate wind power predictions [2]. Physics-based models are commonly represented as a form containing cubic polynomial functions or a power curve of a WT [3]. Although physics-based approaches have the explicit structure and strong interpretability, reported methods might suffer from following drawbacks: 1) explicit parametric models might not have the sufficient flexibility of fully capturing the problem complexity; 2) considered input features are restricted to weather features; and 3) prediction errors of the NWP model might eventually magnify the power prediction errors in several two-stage prediction frameworks.

* Corresponding author.
E-mail address: zijzhang@cityu.edu.hk (Z. Zhang).

https://doi.org/10.1016/j.energy.2020.119356
0360-5442/

Abbreviations

AE	Auto-encoder
AR	Auto-regressive
ARFIMA	Autoregressive Fractionally Integrated Moving Average
BP	Backpropagation
CNN	Convolutional Neural Network
DETL	Deep and Transfer Learning
DNN	Deep Neural Network
EMD	Empirical Mode Decomposition
GPR	Gaussian Process Regression
IAEDNN	Individually developed Auto-encoder-based Deep Neural Network
IRDNN	Individually developed Rectified Linear Unit-based Deep Neural Network
LSTM	Long Short-Term Memory
MAPE	Mean Absolute Percentage Error
MLP	Multilayer Perceptron
MSE	Mean-Square Error
NN	Artificial Neural Network
NWP	Numerical Weather Prediction
PCA	Principal Component Analysis
RE	Reconstruction Error
ReLU	Rectified Linear Unit
RL	Reconstruction Loss
SCADA	Supervisory Control and Data Acquisition
SVR	Support Vector Machine Regression
UDETL	Unified model based on Deep and Transfer Learning
VAR	Vector Auto-regressive
WT	Wind Turbine

In classical data-driven WT power prediction studies, conventional machine learning models and time series models are widely considered. In Ref. [4], support vector machine regression (SVR) and multilayer perceptron (MLP) were applied into the wind power time series and yielded a comparatively satisfactory accuracy. In Ref. [5], authors addressed the error accumulation problem in multi-step power output predictions by using an improved multi-output SVR model. In Ref. [6], a couple of time series models including auto-regressive (AR) model and vector auto-regressive (VAR) model were implemented for multi-step power output predictions. In Ref. [7], effects of meteorological parameters for artificial neural networks (NN) were tested to better tune hyper-parameters of NN and popular machine learning methods, such as the random forest regression, SVR, and Gaussian process regression (GPR), were considered as benchmarks. In Ref. [8], authors proposed an ensemble of machine learning algorithms composed of the decision tree and SVR to yield a higher accuracy and shorten the training time. In Ref. [9], the SVR and autoregressive fractionally integrated moving average (ARFIMA) were conducted to generate linear and non-linear predictions respectively. Although classical data-driven models offer a stronger capability of capturing the nonlinearity and achieve more accurate results, studying more advanced modeling methods for better utilizing data and further improving the prediction accuracy are still desired by the industry.

Recent WT power prediction studies with deep learning target on developing deep models that could progressively extract deep latent features from raw data. Deep models, such as NNs having many hidden layers and a cascade of data-driven models, have been considered as powerful in many studies [10–13]. In Ref. [10], due to the daily similarity of weather conditions, numerical weather data were first denoised through a k-means based clustering analysis and next an NN model was developed based on training data from a same cluster to exploit common latent features for clusters as well as generate predictions. In [11], an ensemble of convolutional neural networks (CNN) were trained to extract deep features for sub-signals decomposed by the wavelet transformation. In Ref. [12], a multi-tasking strategy was adopted to incorporate parameters of multiple WTs in establishing a unified deep neural network (DNN) to generate predictions for WTs. In [13], a CNN-Long Short-Term Memory (LSTM) model was applied to analyze sub-signals of wind power time series decomposed via Empirical Mode Decomposition (EMD). Deep learning has demonstrated a great potential of facilitating wind power predictions. However, studying the paradigm of developing deep NN-based wind power prediction models is still at an early-stage and a direct application of traditional practices in data-driven modeling poses following challenges. First, a general principle for determining the optimal deep NN architecture, especially the number of hidden layers and the number of hidden nodes, for wind power prediction models has not received sufficient discussions. The NN architecture plays a critical role on the modeling capability and training efficiency because too scarce neurons may limit the learning ability of NN while too crowded neurons may incur the overfitting [14]. A large number of trials or empirical knowledge are typically combined to properly determine the network architecture [15–17]. However, to implement deep NN-based methods in wind power predictions via the traditional single task learning, the repetition of the complicated process of developing deep NN-based models over a population of WTs are not user-friendly to wind farm operators as well as result in a large computational burden and high data availability requirement. To enhance the practical value of deep NN-based methods in wind power predictions for multiple WTs, it is important to develop a more efficient computing framework to train deep NN-based wind power prediction models.

The recent development of transfer learning offers an emerging direction to improve the efficiency and accuracy of developing deep NNs as well as address the problem of the lack of data. The transfer learning develops data-driven models via two tasks, source task and target task. Target tasks solves individual learning tasks by using the knowledge from a source task, where the information to be transferred can be parameters, features, instances, etc. [18]. Given a model pre-trained in source task, which utilizes a large volume of data fused from various tasks, the target task then can require less training data and shorter training time. WTs in a wind farm share a similar power generation principle, which can be described by common patterns in their data. Due to different conditions, operation strategies, and locations of WTs, there also exists the heterogeneity among WTs, which can be reflected by individual unique data patterns. Potentially, the source task in transfer learning can be analogous to developing a general model capturing WT homogenous characteristics while the target task can be analogous to customizing models for individual WTs by capturing heterogenous characteristics. To develop NN-based wind power prediction models for multiple WTs under a transfer learning framework, the formulation of source task and target task needs to be investigated. To model two types of characteristics via transfer learning, a new format of organizing SCADA data also needs to be studied. Based on pre-trained model in source task, the transfer learning based modeling framework has a potential of rapidly

producing customized prediction models for a number of WTs. Meanwhile, such procedure might be less bounded by the training data volume for each target task and capable to lift the overall WT modeling efficiency.

Another concern on training DNN models for WTs is the vanishing gradient problem [19], which states that the gradient of first layers would vanish through the backpropagation optimization (BP). It has been suggested that Auto-encoders (AE), as a particular NN structure composed of multi-layers organized symmetrically to reconstruct the input data [20], can serve as a pre-training model to well initialize deep network parameters, such as weights and biases [21]. AE based pre-training has been widely adopted for many NN-based application problems in wind industry [22–25]. Apart from the methodological side, AE itself has successfully proved its unique value in a number of application problems, such as condition monitoring [26], anomaly detection [27,28], and missing data imputations [29] from the application perspective. Therefore, to develop DNNs for predicting wind power via the AE based pre-training, it is also valuable to study a novel approach of more effectively determining AE. Moreover, the integration of transfer learning and AE-based pretraining for developing DNN-based wind power prediction models for a number of WTs has a lack of studies. A related study [17] presented a transfer learning strategy for adapting a DNN ensemble model for wind power predictions of one wind farm to predict wind power of other wind farms, which saved training time. However, the model targeted on a wind farm level power prediction, which was different from wind power predictions at the WT level. In addition, the AE architecture was determined via an exhaustive tuning process and data considered in the source task was from a single wind farm, which might not be sufficient to learn a general model representing common patterns among different wind farms. A framework integrating transfer learning and AE based pre-training, which is able to first extract features describing homogenous characteristics among WT system dynamics and next capture the heterogeneity of each WT, for better developing the DNN-based wind power prediction desired further discussions.

We propose a novel deep and transfer learning (DETL) framework for improving the process of developing wind power prediction models and the accuracy of multi-step power predictions for multiple WTs in a wind farm in this study. Based on a source domain and different target domains for the group of WTs through a data reorganization, the DETL framework training prediction models is composed of three steps: 1) train a base AE for modeling homogenous characteristics in WT dynamics based on the source domain data and the AE architecture is properly determined by a novel adaptive optimization method; 2) specify the base AE model based on the target domain data to model heterogeneity in dynamics of WTs; 3) convert the specified AE models to power prediction models by incorporating additional temporal input features and target variables. Computational studies validated the robust and impressive prediction performance for a group of WTs as well as a higher modeling efficiency offered by our proposed method based on comparing with considered DNN-based benchmarks. Unique contributions of this research include: 1) a more practical method for developing power prediction models of many WTs; 2) a deep and transfer learning based framework for modeling heterogeneous characteristics among WTs is firstly explored; and 3) an adaptive process for optimizing AE architecture is firstly introduced.

The rest of paper is organized as follows. In Section 2, we describe the AE-based WT power prediction model development problem. Section 3 presents the proposed DETL framework. Section 4 reports computational studies, which validate the superiority of the proposed DETL in developing wind power prediction models by benchmarking against considered DNN-based training methods and a physics-based model. Finally, Section 5 concludes this study and presents future works.

2. The deep neural network based wind power prediction problem

In wind power prediction studies considering shallow NNs, relevant SCADA parameters, such as the wind speed, generator torque, blade pitch angle, etc., and the wind power output are separately regarded as inputs and the output of an NN model based on the aerodynamics principle and domain knowledge. SCADA data are then utilized to learn the NN structure for approximating the true complicated WT dynamics. Compared with shallow NNs, developing a high quality DNN for wind power predictions needs to carefully address two issues, properly determining the DNN structure, such as the depth and width, and well handling the vanishing/exploding gradient in training. A two-stage training framework, first un-supervised pre-training with AEs and next supervised fine-tuning by adding model responses, has been widely applied to train high quality DNNs [21–25]. However, the DNN based wind power prediction with such two-stage framework is still not user-friendly to the industrial practitioner because a wind farm contains numerous WTs and developing DNNs for WTs individually from scratches requires repeated modeling efforts, which results in a burden in computations and practices.

To strengthen the applicability of deep learning in wind power prediction, we propose a novel framework, the DETL, which aims to alleviate loads on repeatedly determining DNN models for WTs with the two-stage training framework in the wind power prediction. The overall process of generating wind power predictions for WTs in a wind farm based on the proposed DETL is illustrated in Fig. 1.

Assume a problem scale that wind power outputs of K WTs need to be predicted. As shown in Fig. 1, we first collect historical SCADA data from K WTs to conduct the data-driven model development. We aim to develop K models separately depicting characteristics in dynamics of K WTs to facilitate the wind power prediction. To mitigate computing resources and modeling efforts in developing K prediction models while maintain prediction accuracies, we renovate the AE-based two-stage framework via incorporating an adaptive deep AE development process and the transfer learning, which becomes the proposed DETL.

Let $p_t^{\{i\}}$ denote the wind power output of the i-th WT at time t and T_h denote the prediction horizon, which indicates the total number of prediction steps ahead. Given K prediction models $g^{\{i\}}(\cdot)$, $i = 1, 2, \ldots, K$, developed by the DETL, the multi-step wind power outputs prediction for K WTs can be generally described as $\{p_{t+1}^{\{i\}}, p_{t+2}^{\{i\}}, \ldots, p_{t+T_h}^{\{i\}}\} = g^{\{i\}}(\boldsymbol{x}_t^{\{i\}})$, where $\boldsymbol{x}_t^{\{i\}}$ is a vector of inputs for the i-th WT at the time t. Details of the DETL are presented next.

3. The deep and transfer learning based wind power prediction model development framework

3.1. The description of the DETL framework

To develop data-driven wind power prediction models, we consider SCADA parameters relating to WT power generation process based on domain knowledge and their temporal conditions as possible model inputs. Considered SCADA parameters include the power output, wind speed, pitch angles, temperatures of different WT components, etc. Their temporal conditions are their recent historical records because the future status of a WT could be impacted by historical status according to system dynamics. To well

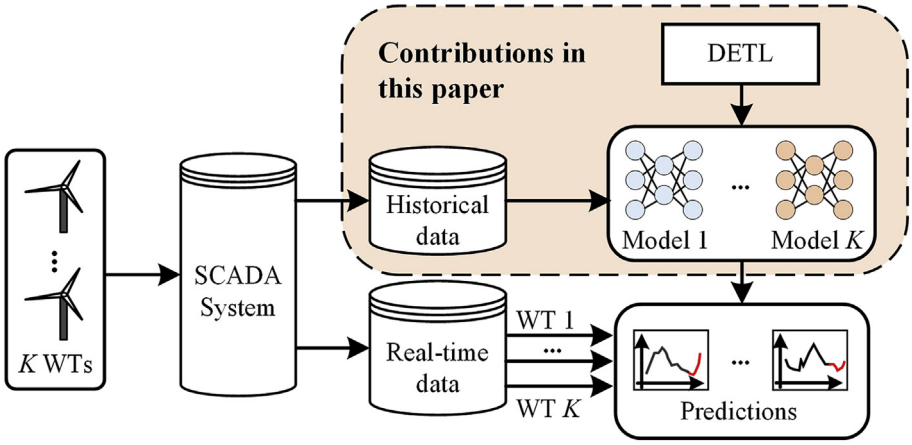


Fig. 1. The process of generating wind power outputs prediction for multiple WTs.

process such high dimensional inputs and develop high quality wind power prediction models, DNNs, which enable the automatic lower dimensional latent feature extraction and offer a stronger capability of modeling complex nonlinear processes, are adopted in this study.

The traditional paradigm, which develops data-driven wind power prediction models from scratches, usually focuses on data of one target WT in previous studies [7,30]. Due to the complexity of training a DNN, directly applying traditional data-driven modeling framework into a commercial wind farm, which contains dozens of WTs, has following challenges. First, determining suitable deep network architectures, such as the number of hidden layers and the number of hidden nodes in each layer, from scratches is computationally extensive and usually needs a large number of experimental trials. The repetition of such modeling process for all WTs apparently further enlarges the computational burden. Meanwhile, obtaining proper initializations of hyper-parameters of multiple deep models, which affects the performance of prediction models, also requires a significant effort. Moreover, the vanishing gradient problem in first hidden layers [19] may hinder the DNN training process. Therefore, to avoid the repetition of the complicated modeling process and enhance the ease-of-use of deep learning in wind power predictions, we adopt a transfer learning scheme in the proposed DETL framework as shown in Fig. 2, which is able to save the modeling effort and generate higher quality models.

As shown in Fig. 2, the transfer learning scheme in DETL is described by three steps. Considering that K WTs share same generation principle and they are of similar types, relationships between wind power, wind speed, rotor speed, and temperature, etc. have the similarity among WTs, which can be regarded as homogeneous patterns. Meanwhile, due to the wake effect, conditions of WT sub-systems, control strategies, etc., power generation processes of K WTs also present differences and demonstrate heterogeneous characteristics. To efficiently model dynamics of each WT, the transfer learning process is developed to establish K individual models, which in particular first develops a base model for representing the homogeneous patterns of all WTs and next specializes the base model to ones depicting the heterogeneity among WTs. Finally, temporal features indicating the wind power dynamics are incorporated into K models developed to supplement the temporal information. Meanwhile, target variables, multi-step ahead wind power outputs, are added into the output layer of K models to generate predictions. The proposed DETL, which adopts transfer learning, and the traditional method, which utilizes the single task learning, in developing prediction models for multiple WTs, are compared in Fig. 3.

Based on the left part of Fig. 3, the traditional single task learning repeats the development of the power prediction model for each WT from scratches and each model is developed based on data of the corresponding WT only. Such paradigm might not sufficiently model homogeneous patterns among all WTs. Considering the significant time consumed in determining K DNN architectures and optimizing network parameters, the total computational burden is large and the whole process sounds quite complicated. Yet, in DETL, as shown in the right part of Fig. 3, a base model describing homogeneous patterns among WTs via a source domain learning is developed based on a collection of sub-datasets of all WTs. Next, wind power prediction model customized for each WT is developed by extending the base model based on a small amount of SCADA data of the corresponding WT via a target domain learning. In such process, a well determined base model can serve as a foundation for producing tailor-made wind power prediction models for different WTs so that a significant amount of modeling efforts can be saved.

To more explicitly illustrate the full picture of the DETL method, the complete process is visualized as a flowchart in Fig. 4. The main algorithm after preparing data is described in **Algorithm 1**. In following subsections from Sections 3.2 to 3.5, blocks in Fig. 4 and **Algorithm 1** are explained in details.

Algorithm 1
DETL

Input:
Source domain training set D_{st};
Source domain validation set D_{sv};
Fine-tuning sets $D_{tf}^{\{1\}}$, $D_{tf}^{\{2\}}$, …, $D_{tf}^{\{K\}}$;
Threshold of terminating the hidden layer augmentation θ_1;
Threshold of terminating the hidden unit augmentation θ_2;
Step size of adding new hidden units at each update λ;
Length of lag features T_l;
Length of prediction horizon T_h;
Output:
AE and DNN architectures $\{L, \boldsymbol{N}_f, \boldsymbol{N}_g\}$;
Weights and biases of DNNs for modeling WTs $\{\boldsymbol{W}_g^{\{i\}}, \boldsymbol{B}_g^{\{i\}}\}$, $i = 1, 2, \ldots, K$;
A) Train a self-organizing AE with the source domain by:
$\{\boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\} = SelfOrganAE(D_{st}, D_{sv}, \theta_1, \theta_2, \lambda)$; $L = |\boldsymbol{B}_f^{\{0\}}| - 1$;
$\boldsymbol{N}_f = \{N_{f,i} | N_{f,i} = \dim(B_{f,i}^{\{0\}}), i = 1, \ldots, L{+}1$, while $N_{f,0} = r\}$;
$\boldsymbol{N}_g = \{N_{g,i} | N_{g,i} = N_{f,i}, i = 1, \ldots, L$, while $N_{g,0} = r + T_l, N_{g,L+1} = r + T_h\}$;
B) Tune the pre-trained DNN in each target domain:
For $i = 1$ to K do:
$\{\boldsymbol{W}_f^{\{i\}}, \boldsymbol{B}_f^{\{i\}}\} = BPOptimization(\boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}, D_{tf}^{\{i\}})$;
C) Transform the pre-trained AE to DNN prediction model:
$\{\boldsymbol{W}_g^{\{i\}}, \boldsymbol{B}_g^{\{i\}}\} = TransAEtoDNN(\boldsymbol{W}_f^{\{i\}}, \boldsymbol{B}_f^{\{i\}}, D_{tf}^{\{i\}}, T_l, T_h)$;
End For

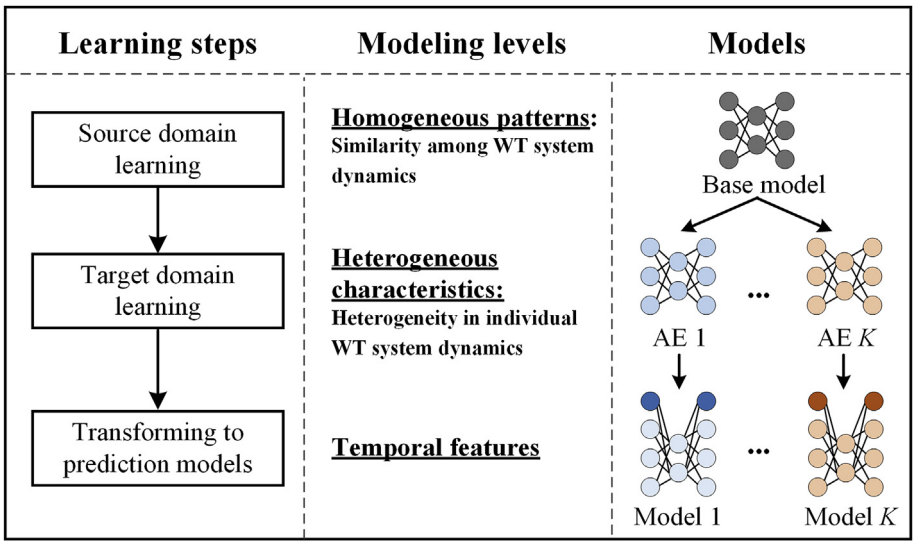


Fig. 2. A transfer learning scheme in DETL.

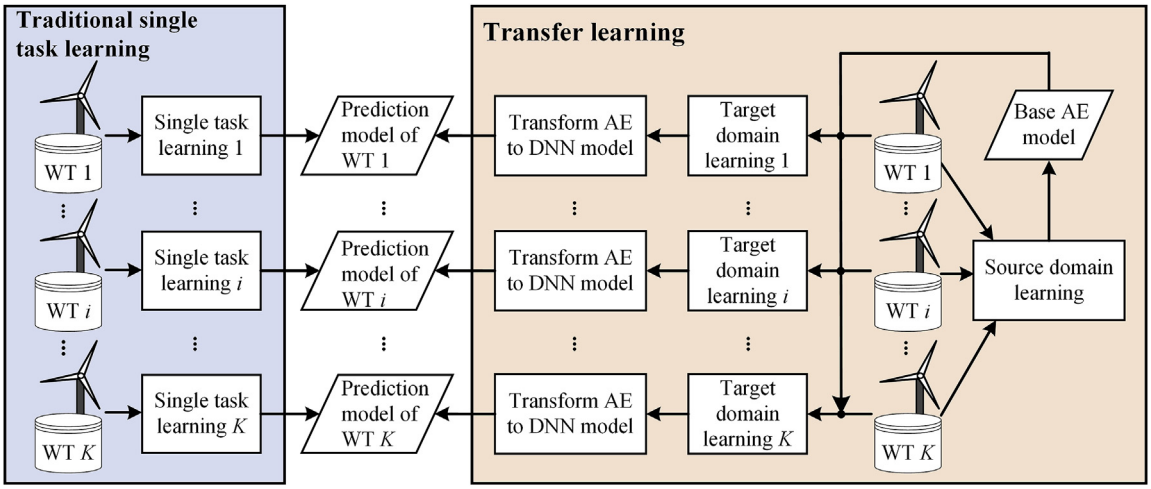


Fig. 3. The modeling framework comparison between traditional single task learning and transfer learning.

3.2. Data reorganization

To enable a transfer learning process first extracting homogeneous patterns among all WTs and next specializing heterogeneity among dynamics of WTs, the source domain and target domains need to be firstly constructed, which refers to the *Rearrange data and set hyper-parameters* block in Fig. 4.

Assume that SCADA data of K WTs, denoted as $D^{\{i\}}$, $i = 1, 2, \ldots, K$, are available. Suppose that $D^{\{i\}}$, $i = 1, 2, \ldots, K$, has the identical r SCADA parameters including the power output p and can be further expressed as $D^{\{i\}} = \{\mathbf{X}_1^{\{i\}}, \mathbf{X}_2^{\{i\}}, \ldots, \mathbf{X}_M^{\{i\}}\}$, where M means the total number of data points in $D^{\{i\}}$. The source domain and target domains can be constructed according to the Definition 1 and Definition 2, respectively.

Definition 1. Let $\widehat{D}^{\{i\}}$ denote a set of a q proportion of data points from $D^{\{i\}}$. A source domain, D_s, is a union of $\widehat{D}^{\{i\}}$ for $i = 1$ to K, $D_s = \cup_{i=1}^{K} \widehat{D}^{\{i\}}$.

Definition 2. Let $D'^{\{i\}}$ denote a subset from $D^{\{i\}}$ excluding $\widehat{D}^{\{i\}}$, $i = 1, 2, \ldots, K$. A target domain of the i-th wind turbine, $D_t^{\{i\}}$, is $D'^{\{i\}}$ or a subset of $D'^{\{i\}}$.

A possible data rearrangement is presented in Fig. 5. The $D^{\{i\}}$, $i = 1, 2, \ldots, K$, is separated to form a source domain and K target domains according to Definitions 1 and **2**. The source domain is further divided into a source domain training set, D_{st}, with M_{st} data points and a source domain validation set, D_{sv}, with M_{sv} data points. Since the base model can provide a decent parameter initialization, it is unnecessary to apply a large amount data to adjust the base model in the target domain learning, while if there are sufficient remaining data, we can consider just using a part of them to save the training time. The target domain of the i-th WT is split into a fine-tuning set, $D_{tf}^{\{i\}}$, and a test set, $D_{tt}^{\{i\}}$, for $i = 1, 2, \ldots, K$. Data split ratios in Fig. 5 are applied into computational studies of this research. Split ratios are adjustable due to goals of different applications.

3.3. The base model development

After data are rearranged, homogeneous patterns among all

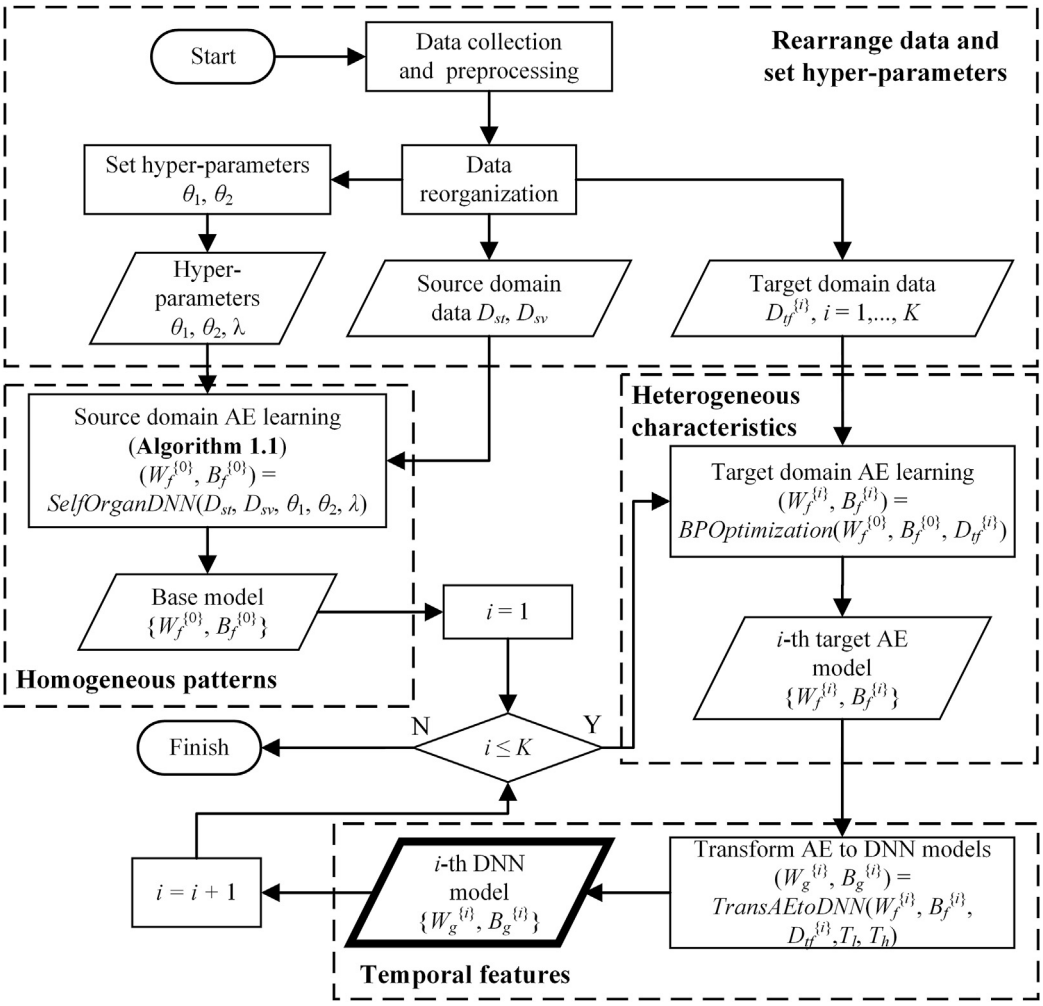


Fig. 4. The DETL framework for establishing wind power prediction models for multiple WTs.

WTs could be extracted and modeled by a base model. We consider applying the AE model to extract homogeneous patterns for three reasons. First, the AE model is a well-known unsupervised learning model by reconstructing the input data accurately and its latent layer is usually interpreted as hidden representations of the input space. Second, as we study wind power predictions with NN-based models, AE models can be easily transformed to supervised NN-based prediction models. Moreover, AE models are often considered as a pre-training of DNNs to address the vanishing gradient problem. In the development of AE models, we simply use all available non-constant SCADA parameters, e.g. the wind power, wind speed, rotor speed, temperatures, etc., as model inputs to comprehensively describe the WT system.

Training the base AE model can be formulated as an optimization problem. Let $f^{\{0\}}(\cdot)$ denote the base AE model of all WTs. The $f^{\{0\}}(\cdot)$ is parameterized by a set of parameters $\{L, \boldsymbol{N}_f, \boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\}$, where the subscript f refers to AE models, L refers to the total number of hidden layers, $\boldsymbol{N}_f = \{N_{f,0}, N_1, \ldots, N_L, N_{f,L+1}\}$ means the number of nodes of all layers with $N_{f,0} = N_{f,L+1} = r$ indicating the number of nodes in the input and output layers respectively, as well as $\boldsymbol{W}_f^{\{0\}} = \{W_{f,1}^{\{0\}}, W_{f,2}^{\{0\}}, \ldots, W_{f,L+1}^{\{0\}}\}$ and $\boldsymbol{B}_f^{\{0\}} = \{B_{f,1}^{\{0\}}, B_{f,2}^{\{0\}}, \ldots, B_{f,L+1}^{\{0\}}\}$ present weights and biases distributed across connections among all layers of $f^{\{0\}}(\cdot)$. Solving such modeling problem is equivalent to minimizing a reconstruction loss (RL), $J = \|\mathbf{X} - \widehat{\mathbf{X}}\|^2$, where $\widehat{\mathbf{X}}$ denotes the feature space reconstructed by the model $f(\cdot)$ developed by a data-driven method using source domain training data D_{st}. Based on RL, we define a reconstruction error (RE), $R = \frac{1}{M}\sum_{\forall j}\|\mathbf{X}_j - \widehat{\mathbf{X}}_j\|^2$, where $\widehat{\mathbf{X}}_j$ denotes the j-th reconstructed data point of D_{st}, to evaluate the model performance in a form of mean-square error (MSE). Therefore, the development of $f^{\{0\}}(\cdot)$ can be translated to an optimization problem which optimizes $\{L, \boldsymbol{N}_f, \boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\}$ through minimizing J.

The network architecture of the base AE is represented by values of L and $\boldsymbol{N}_f$ and dimensions of $\boldsymbol{W}_f^{\{0\}}$ and $\boldsymbol{B}_f^{\{0\}}$. To optimize the network architecture, although a few WT-related studies proposed to drop out less important connections of an initialized NN [31,32], the method of determining a suitable depth of a NN remained unclear. To enable an automated architecture optimization of an AE network, we develop an algorithm for self-expanding the network through continuously stacking hidden layers and units until a suitable size is achieved. This strategy has three significant merits: 1) the network parameters that have been optimized in previous updates are retained and inherited to next updates, which could avoid wasting the training outcomes, compared to the traditional trial-and-errors which sets several independent AEs and trains them completely from scratch; 2) The learning ability of the model is guaranteed to enhance during the optimization by adding new elements to the network. 3) By setting the termination criterion to a continuous increase of validation errors for a certain number of times, overfitting can be prevented.

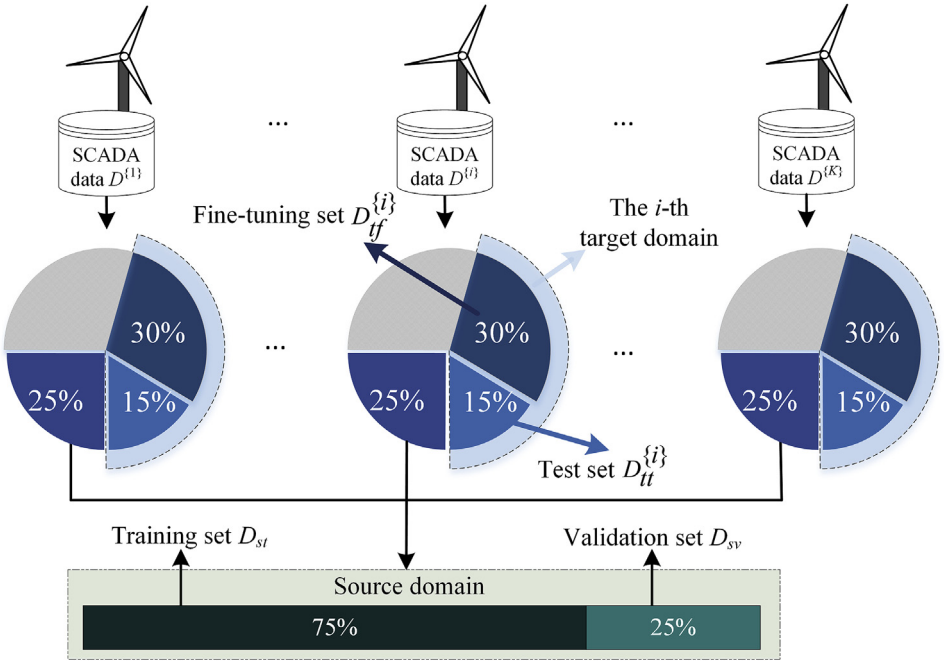


Fig. 5. The data rearrangement for studying the DETL framework.

Based on the source domain data D_{st} and D_{sv} and three hyper-parameters, θ_1 and θ_2, which are coupled with the termination criterion, as well as λ controlling the step size of increasing network architecture, $\{L, \boldsymbol{N}_f, \boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\}$ can be automatically optimized. The proposed source domain training method, *SelfOrganAE*, is briefly depicted by the Pseudo Code, **Algorithm 1.1**.

Algorithm 1.1
SelfOrganAE

Input:
Source domain training set D_{st};
Source domain validation set D_{sv};
Threshold of terminating the hidden layer augmentation θ_1;
Threshold of terminating the hidden unit augmentation θ_2;
Step size of adding new hidden units at each update λ;
Output:
Weights and biases of the base AE $\{\boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\}$;
(A) Initialize parameters: $l = 0$, $\boldsymbol{W}_l = \{\}$, $\boldsymbol{B}_l = \{\}$;
(B) While $\{(l > \theta_1)$ and $(R_v^{\{l\}} \geq R_v^{\{l-1\}} \geq \ldots \geq R_v^{\{l-\theta_1\}}$ is not True)$\}$ or $(l \leq \theta_1)$:
(B-1) Add a new hidden layer l: $\{W_e, W_d, B_e, B_d\} = LayerOptimization(D_{st}, D_{sv}, l, \boldsymbol{W}_l, \boldsymbol{B}_l, \theta_2, \lambda)$;
(B-2) $l = l+1$;
(B-3) Update DNN:
If $l = 1$ do: $\boldsymbol{W}_l = \{W_e, W_d\}$, $\boldsymbol{B}_l = \{B_e, B_d\}$;
Else do: $\boldsymbol{W}_l = \{W_{(l-1,1)}, \ldots, W_{(l-1,\ l)}, W_e, W_d\}$, $\boldsymbol{B}_l = \{B_{(l-1,1)}, \ldots, B_{(l-1,\ l)}, B_e, B_d\}$;
End If;
(B-4) Compute RE: $\widehat{\mathbf{X}} = feedforward(\boldsymbol{W}_l, \boldsymbol{B}_l, D_{sv})$;

$$R_v^{\{l\}} = \frac{1}{M_{sv}} \sum_{\mathbf{x}_j \in D_{sv}} \left\| \mathbf{x}_j - \widehat{\mathbf{x}}_j \right\|^2;$$

End While;
(C) $L = \underset{l}{\operatorname{argmin}}(R_v^{\{l\}})$;
(D) Tune DNN parameters:
$(\boldsymbol{W}^{\{0\}}, \boldsymbol{B}^{\{0\}}) = BPOptimization(\boldsymbol{W}_L, \boldsymbol{B}_L, D_{st})$;

Let $\boldsymbol{W}_l$ and $\boldsymbol{B}_l$ denote weight and bias sets of an AE with l hidden layers while $W_{(l,\ i)}$ and $B_{(l,\ i)}$ denote the i-th weight and bias matrices in $\boldsymbol{W}_l$ and $\boldsymbol{B}_l$. As illustrated in Fig. 6, the *SelfOrganAE* considers an operation which iteratively creates a new hidden layer $l+1$ between the hidden layer l and the output layer. In such operation, all elements of $\boldsymbol{W}_l$ and $\boldsymbol{B}_l$ except $W_{(l,\ l+1)}$ and $B_{(l,\ l+1)}$ are firstly inherited to $\boldsymbol{W}_{l+1}$ and $\boldsymbol{B}_{l+1}$. Meanwhile, weights, $\{W_e, W_d\}$, and biases, $\{B_e, B_d\}$, produced due to the insertion of a new hidden layer are also included into $\boldsymbol{W}_{l+1}$ and $\boldsymbol{B}_{l+1}$, where e stands for the encoder and d stands for the decoder. Given weights, $\{W_1, \ldots, W_l\}$, and biases, $\{B_1, \ldots, B_l\}$, the *SelfOrganAE* next optimizes $\{W_e, W_d, B_e, B_d\}$ based on the source domain training set D_{st} by an algorithm *LayerOptimization* as described in **Algorithm 1.1.1**, to minimize the domain training RL, denoted as $J_t^{\{l+1\}}$. After the optimization of $\{W_e, W_d, B_e, B_d\}$, the D_{sv} is fed into the AE to estimate RE under a source domain validation, denoted as $R_v^{\{l+1\}}$. The stopping criterion of adding additional hidden layers is set to that R_v continuously increases for θ_1 times after stacking additional hidden layers. Finally, the structure, $\{L, \boldsymbol{N}_f\}$, leading to a minimal R_v is selected and corresponding weight and bias sets, $\{\boldsymbol{W}_L, \boldsymbol{B}_L\}$, are tuned via BP based on the source domain to obtain $\{\boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\}$.

The optimization of generated weights, $\{W_e, W_d\}$, biases, $\{B_e, B_d\}$, and the number of hidden units on the $l+1$-th hidden layer, N_{l+1}, based on the *LayerOptimization* in **Algorithm 1.1.1** follows a similar iterative search principle for stacking hidden layers. Given a hidden layer, the number of hidden units are continuously increased until a stopping criterion that $R_v^{\{l+1\}}$ continuously grows for θ_2 times is met. Next, weights and biases are finalized with the determined number of hidden units. Details of such procedure are illustrated in Fig. 7.

In the *LayerOptimization*, N_{l+1} is determined by iteratively adding additional hidden units into the hidden layer $l+1$ until θ_2 times of continuous growth of $R_v^{\{l+1\}}$ is met. Thus, the initial value of N_{l+1} needs to be appropriately determined. Ideally, N_{l+1} should be initialized to 1 to enable an exhaustive search which, however, can be time-consuming and inefficient. Since the AE structure will be expanded in the DETL, N_{l+1} can be initialized as a relatively large value but not greater than the potential optimal value to improve the efficiency. Referring to Ref. [26], a dimension reduction, the Principal Component Analysis (PCA) [33], is applied based on the output of the hidden layer l, H_l, to obtain the initial value for N_{l+1}. It

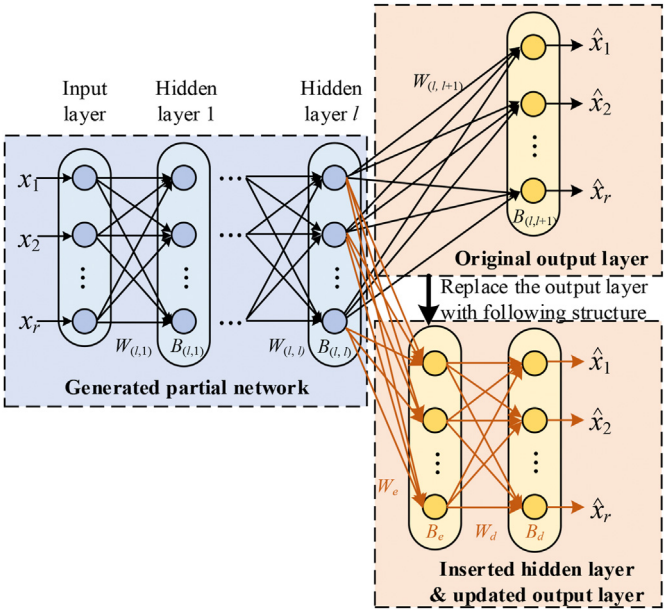


Fig. 6. Optimization process of adding a hidden layer.

is based on an assumption that the linear relationship between features is the simplest relationship which may not require more hidden units as learning non-linear relationship. However, as a linear dimension reduction technique, PCA only serves the initialization of the value of N_{l+1} and is not applied to learn weights on new connections created.

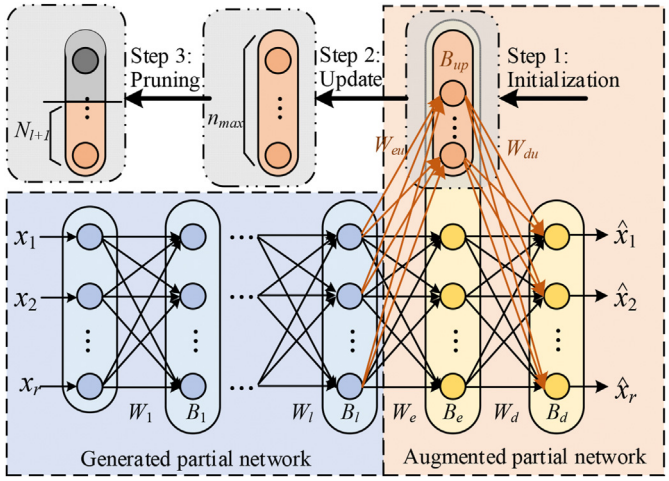


Fig. 7. Optimization process of determining hidden nodes.

Details of the *LayerOptimization* are described in following three steps:

Step 1. **Initialization:** Compute H_l by feeding D_{st} into the original AE via a feedforward algorithm. The initial value of N_{l+1} is determined with the help of PCA. First, the number of the greatest eigenvalues occupying more than 80% of the sum of all eigenvalues is obtained. Next, we set it as the initial number of hidden units in the hidden layer l+1. Create connections among the l-th hidden layer, the l+1-th hidden layer, and the output

Algorithm 1.1.1
LayerOptimization

Input:
Source domain training set D_{st};
Source domain validation set D_{sv};
Total number of hidden layers in the current AE l;
A set of weight matrices before update $\boldsymbol{W}_l$;
A set of bias matrices before update $\boldsymbol{B}_l$;
Threshold of terminating the hidden unit augmentation θ_2;
Step size of adding new hidden units at each update λ;
Output:
The weights connected to the inserted hidden layer W_e, W_d;
The biases of the inserted hidden layer and the output layer B_e, B_d;
(A) Initialization: denote the number of current iteration as t, initialize $t = 0$;
(A-1) If $l = 0$ do: Apply PCA on D_{st} to reduce the dimension from r to n as an initialization;
Else do: $H_l = feedforward(\{W_{(l,1)}, \ldots, W_{(l,l)}\}, \{B_{(l,1)}, \ldots, B_{(l,l)}\}, D_{st})$, apply PCA on H_l to reduce the dimension from N_l to n as an initialization;
End If
(A-2) Randomly initialize $W_e \in \mathbb{R}^{N_l \times n}$, $W_d \in \mathbb{R}^{n \times r}$, $B_e \in \mathbb{R}^n$, $B_d \in \mathbb{R}^r$;
(B) Update hidden units: While {($t > \theta_2$) and ($R_v^{\{t\}} \geq R_v^{\{t-1\}} \geq \ldots \geq R_v^{\{t-\theta 1\}}$ is not True)} or ($t \leq \theta_2$): or ()
(B-1) Randomly initialize $W_{eu} \in \mathbb{R}^{N_l \times \lambda}$, $W_{du} \in \mathbb{R}^{\lambda \times r}$, $B_{eu} \in \mathbb{R}^{\lambda}$;
(B-2) Update parameters: $W_e \leftarrow [W_e\ W_{eu}]$, $W_d \leftarrow \begin{bmatrix} W_d \\ W_{du} \end{bmatrix}$, $B_e \leftarrow \begin{bmatrix} B_e \\ B_{eu} \end{bmatrix}$, $t \leftarrow t + 1$;
(B-3) Optimize W_e, W_d, B_e, B_d: $(\{W_e, W_d\}, \{B_e, B_d\}) \leftarrow BPOptimization(\{W_e, W_d\}, \{B_e, B_d\}, H_l)$;
(B-4) Compute RE: $\widehat{\mathbf{X}} = feedforward(\{W_{(l,1)}, \ldots, W_{(l,l)}, W_e, W_d\}, \{B_{(l,1)}, \ldots, B_{(l,l)}, B_e, B_d\}, D_{sv})$;

$$R_v^{\{t\}} = \frac{1}{M_{sv}} \sum_{\mathbf{X}_j \in D_{sv}} \left\| \mathbf{X}_j - \widehat{\mathbf{X}}_j \right\|^2;$$

End While;
(C) Pruning:
(C-1) $T = \underset{t}{\operatorname{argmin}}(R_v^{\{l+1,t\}})$, $N_{l+1} = n + \lambda T$;
(C-2) Only retain the first N_{l+1} columns of W_e, and the first N_{l+1} rows of W_d and B_e;
(C-3) Tune the parameters: $(\{W_e, W_d\}, \{B_e, B_d\}) \leftarrow BPOptimization(\{W_e, W_d\}, \{B_e, B_d\}, H_l)$;

layer. Next, randomly initialize $W_e \in \mathbb{R}^{N_l \times n}$, $W_d \in \mathbb{R}^{n \times r}$, $B_e \in \mathbb{R}^n$, $B_d \in \mathbb{R}^r$.

Step 2. **Hidden unit update:** Iteratively add λ new hidden units into the l+1-th hidden layer and evaluate REs using the validation set, where $\lambda \in \mathbb{N}^+$ is the step size of adding new hidden units at each update. Let $W_{eu} \in \mathbb{R}^{N_l \times \lambda}$, $W_{du} \in \mathbb{R}^{\lambda \times r}$, and $B_{eu} \in \mathbb{R}^{\lambda}$ denote weights of connecting the l-th hidden layer and λ hidden units, weights of connecting λ hidden units and the output layer, as well as the biases of λ hidden units. At each update, W_{eu}, W_{du} and B_{eu} are randomly initialized and aggregated into the W_e, W_d and B_e, respectively. The W_e, W_d, B_e and B_d will be optimized using the gradient descent while fixing parameters from the input layer to the hidden layer l. Next, RE estimated using the validation set will be computed and denoted as $R_V^{\{t\}}$ for the hidden layer at the t-th update. The update process should terminate when $R_V^{\{t\}}$ continues to increase θ_2 times or optionally the number of units reaches a predefined limit.

Step 3. **Pruning and tuning:** By comparing $R_V^{\{t\}}$, T is set to an t which can result in the minimal $R_V^{\{t\}}$. The corresponding number of hidden units can be computed by $N_{l+1} = n + \lambda T$. Retain the first N_{l+1} hidden units and prune the rest. Update W_e, W_d and B_e to keep the corresponding rows or columns of the matrices. Finally, apply the BP to further optimize the W_e, W_d, B_e and B_d.

Homogeneous patterns among WTs can be represented by the L-th hidden layer of the developed base model $f^{\{0\}}(\cdot)$.

3.4. Modeling heterogeneity among WTs

The base AE model is insufficient to accurately describe the heterogeneity among WTs due to different conditions of WTs, wake effects, experienced maintenance activities, etc. To specialize unique characteristics of each WT, K individual AE models well describing heterogeneity among K WTs respectively are further developed. To utilize homogeneous patterns of all WTs extracted via the source domain learning, the base model $f^{\{0\}}(\cdot)$ is regarded as an initialization of individual models by providing the network architecture and values of network weights and biases. Let $f^{\{i\}}(\cdot)$ denote the AE model of the i-th WT, which is parameterized by a set of parameters $\{L, \boldsymbol{N}_f, \boldsymbol{W}_f^{\{i\}}, \boldsymbol{B}_f^{\{i\}}\}$. Transforming $\{\boldsymbol{W}_f^{\{0\}}, \boldsymbol{B}_f^{\{0\}}\}$ to $\{\boldsymbol{W}_f^{\{i\}}, \boldsymbol{B}_f^{\{i\}}\}$ can be efficiently realized through a BP optimization based on target domain data $D_U^{\{i\}}$, $i = 1, 2, \ldots, K$. The learning process in **Algorithm 1** is denoted as *BPOptimization*.

3.5. Temporal feature incorporation and prediction generations

Developed AE models focus on modeling physical characteristics of WTs. To facilitate the wind power prediction, temporal characteristics reflecting the dynamics of wind power time series also need to be considered and incorporated into prediction models.

Assume that inputs of developed AE models are SCADA parameters at time t, which is denoted by $\{x_1, \ldots, x_r\}$, and we aim to predict power outputs from t+1 to $t + T_h$. To study the temporal relationship as well as utilize latest available data, lagged power outputs from p_{t-T_l} to p_{t-1} are supplemented into the input layer as input features, where T_l is the lag length. Meanwhile, to convert AE models to prediction models, target variables, $p_{t+1}, p_{t+2}, \ldots, p_{t+T_h}$, are added into the output layer. By adding input features and target variables, new connections and units are created, as represented by red arrows in Fig. 8, as well as weights and biases are attached. Therefore, based on the network parameters inherited from the developed AE model, a DNN prediction model can be initialized by incorporating several newly created parameters, and finalized through a fast fine-tuning with target domain data.

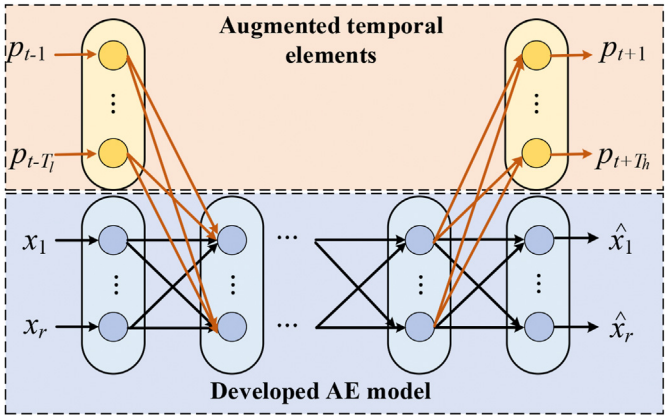


Fig. 8. Transformation of an AE model to a DNN prediction model.

Compared to the traditional paradigm of transforming an AE to a DNN which prefers to directly cascade the target variables to the hidden layers and discard the nodes and parameters related to reconstruction [22,23], the proposed method changes the model to a multi-tasking architecture and retains the elements for reconstruction. The loss induced by the reconstruction error is expected to serve as a penalty term for regularization, which may help prevent the latent representation of learnt physical characteristics in AE training from a drastic shifting and overfitting.

Let $g^{\{i\}}(\cdot)$ denote the multi-step wind power prediction model of the i-th WT parameterized by $\{L, \boldsymbol{N}_g, \boldsymbol{W}_g^{\{i\}}, \boldsymbol{B}_g^{\{i\}}\}$. According to previous descriptions, the $\boldsymbol{N}_g$ and $\boldsymbol{N}_f$ should differ only in the first element and the last element, $N_{g,0} = N_{f,0} + T_l$ and $N_{g,L+1} = N_{f,L+1} + T_h$. The model input $\boldsymbol{x}$ is the vector $\{p_{t-T_l}, \ldots, p_{t-2}, p_{t-1}, x_1, \ldots, x_r\}$. Network parameters $\{\boldsymbol{W}_g^{\{i\}}, \boldsymbol{B}_g^{\{i\}}\}$ are optimized by the BP method from initializations $\{\boldsymbol{W}_f^{\{i\}}, \boldsymbol{B}_f^{\{i\}}\}$. The transformation from $f^{\{i\}}(\cdot)$ to $g^{\{i\}}(\cdot)$ is represented by *TransAEtoDNN* in **Algorithm 1**.

4. Computational experiments

In this section, we demonstrate and verify advantages of the DETL in developing wind power output prediction models through the prediction accuracy and training efficiency obtained from computational experiments. To evaluate the performance of DETL considering different prediction horizons as well as the number of WTs and wind farms, two sets of experiments, in which one set targets on 10-s to 1-min ahead six-step wind power predictions for 15 WTs and another set assesses 10-min to 1-h ahead six-step predictions for 50 WTs, are conducted and results are analyzed.

4.1. Data description and experimental settings

SCADA data considered in computational experiments were collected from 18 WTs with a rated power of 1.5 MW in the commercial wind farm A and 32 WTs with a rated power of 1.6 MW in the commercial wind farm B. Data descriptions are briefly

Table 1
SCADA data descriptions of WTs in wind farms A and B.

	Wind farm A	Wind farm B
Sampling interval	10-s	10-min
Number of SCADA parameters	73	30
Number of WTs	18	32
Time	Aug–Dec, 2010	May–Aug, 2015

Table 2
Settings and data utilization in experiments.

	Experiment A	Experiment B
WTs utilized	15 WTs from the wind farm A	50 WTs from wind farms A and B
Interval of each step prediction	10-s	10-min
Prediction horizon	From 10-s to 60-s	From 10-min to 60-min
Number of SCADA parameters	73	22

Table 3
The 22 SCADA parameters utilized in Experiment B.

No.	Parameter	No.	Parameter
1	Wind power output	12	Generator temperature 2
2	Wind speed	13	Voltage (phase A)
3	Nacelle position	14	Voltage (phase B)
4	Generator speed	15	Voltage (phase C)
5	Outside temperature	16	Current (phase A)
6	Wind deviation	17	Current (phase B)
7	Pitch angle (blade 1)	18	Current (phase C)
8	Pitch angle (blade 2)	19	Rotor speed
9	Pitch angle (blade 3)	20	Gearbox temperature
10	Nacelle temperature	21	Bearing A temperature
11	Generator temperature 1	22	Bearing B temperature

summarized in Table 1.

Both datasets include key parameters in WT power generations, such as the power output, the voltage and current, rotational speeds of mechanical components, temperatures at various locations, etc. Based on data of 50 WTs, details of two sets of aforementioned experiments, Experiment A and Experiment B, are further described in Table 2.

In Table 2, Experiment A is conducted to validate the effectiveness of DETL when more homogenous patterns exist among WTs. Therefore, a small number of WTs of the same type from the same farm are selected, and their training data were collected over an identical period. A shorter prediction horizon, six steps ahead predictions from 10-s to 60-s, is studied in the Experiment A. Three WTs in the wind farm A are not included in the Experiment A because they have comparatively more missing records and the time-stamp of their data sampling is not consistently aligned with other 15 WTs. In other WTs, records containing missing data are simply excluded.

In Experiment B, a more generalized scenario that wind power productions of a large number of WTs show more heterogeneous characteristics is studied in predictions. Data of WTs in different wind farms collected at different time are utilized. To enable a unified modeling for WTs across two wind farms, 22 SCADA parameters presented in Table 3 obtained via intersecting sets of raw SCADA parameters of all WTs are selected as model inputs. Multi-step wind power predictions over a longer prediction horizon from 10-min to 60-min are conducted in Experiment B.

To train NN models effectively, all data are scaled to the range [0, 1]. In one-to six-step ahead prediction tasks, p_{t+1}, …, p_{t+6} should be predicted. The length of lagged time series features is set to 30 steps, i.e., $p_{t\text{-}29}$, …, p_t. They are considered as temporal input features as well as separately present a total lag of 5-min in Experiment A and 5-hrs in Experiment B.

4.2. Benchmarking methods

To validate the accuracy and efficiency of the DETL modeling process, one physics-based benchmarking model and three deep learning benchmarking models, in which two of them separately develop the wind power prediction models based on the dataset of corresponding WT only and the other develops a unified model with all data collected from all WTs, are considered.

The first modeling approach, the IRDNN, in this paper refers to individually developing DNN-based prediction models for each WT with an activation function, the Rectified Linear Unit (ReLU). Since the ReLU activation function can impose a sparsity in the network, an AE-based pre-training is exempted and the overfitting can be prevented [34]. Moreover, the optimization of network is computationally efficient due to the simple form of gradients. Therefore, IRDNN is often considered as a powerful benchmark.

The second benchmark, the IAEDNN, refers to individually developing DNN-based prediction models trained from AE models for each WT in a conventional way. In IAEDNN, an AE model is first trained with data of the corresponding WT to extract features and initialize network parameters. Next, the output layer of the AE is replaced with the target variables, p_{t+1}, …, p_{t+6}, to produce the DNN prediction model. The IAEDNN is a very common modeling approach for tackling the vanishing gradient problem.

The third benchmark is a physics-based model, called PM in this paper, which predicts the power outputs through computing power coefficients. The physics-based model of each WT is trained based on its data only. According to Ref. [35], the wind power generation model can be written as (1)

$$p = \frac{1}{2}\rho\pi R^2 C_p(\lambda,\beta)v^3 \tag{1}$$

where p is the wind power output, ρ is the air density, R is the rotor radius, C_p is the power coefficient, which is impacted by the tip speed ratio λ and blades pitch angle β, as well as v is the wind speed. The tip speed ratio is computed by $\lambda = \frac{\omega_r R}{v}$, where ω_r is the rotor speed. To predict the wind power based on the physics-based model, following steps are performed:

Step 1. Compute C_p based on training data using (2) which is derived from (1) when all other parameters in (1) are known;

$$C_p = \frac{2p}{\rho\pi R^2 v^3} \tag{2}$$

Step 2. Fit $C_p(\lambda,\beta)$ into data with a polynomial function, whose order is determined through experimental trials [34];
Step 3. To avoid the bias caused by an inaccurate NWP, we assume an ideal scenario that actual values of all SCADA parameters and wind speed at t+1 can be accurately obtained in advance. Predict C_p for the test set via $\tilde{C}_p = C_p(\lambda_{t+1},\beta_{t+1})$;
Step 4. Perform wind power predictions by (3).

Table 4
REs of the DETL in source domain learning in Experiment A.

AE depth	AE structure	RE on validation set (× 0.001)
1	**73 - 100 - 73**	**0.413**
2	73 - 100–69 - 73	0.706
3	73 - 100–69 - 49 - 73	1.504

$$\tilde{P}_{t+1} = \frac{1}{2}\rho\pi R^2 \tilde{C}_p v_{t+1}^3 \tag{3}$$

Considering an ideal assumption that the natural conditions and wind speed in short future are known, the multi-step predictions are essentially equivalent to one-step prediction. Hence, in PM, we only perform the one-step ahead prediction. However, the ideal assumption of known natural conditions in 10-min is not as realistic as in 10-s. Thus, the PM is only conducted in Experiment A rather than in Experiment B. In Experiment A, the rotor radius of WTs from the wind farm A is available, $R = 38.5$ m, the air density is assumed as a constant $\rho = 1.29$ kg/m^3, and β is obtained by the average of three blades pitch angles, $\beta = (\beta_1 + \beta_2 + \beta_3)/3$.

To demonstrate the existence of heterogeneous characteristics of WTs and the necessity of developing separate prediction models for the group of WTs, the last benchmarking model, the UDETL, is implemented, which develops a unified prediction model for all WTs. The UDETL performs a source domain learning to train an AE model using training data collected from all WTs with an adaptive architecture determination mechanism as introduced in DETL and transforms the AE model to a DNN prediction model without an intermediate target domain learning.

4.3. Implementation details of the DETL

In DETL, the AE-based model structure is not pre-defined but automatically determined through the training process. The total training epochs are unknown in advance due to the greedy stopping criteria governed by θ_1 and θ_2. Large values of θ_1 and θ_2 will be beneficial to avoid the local optima traps; however, they might lead to excessive redundant training epochs. Low values of θ_1 and θ_2 can cause an early termination of the training process to save the training time; however, they might make the training process too greedy to bounce from the local optima. Similarly, low values of λ may lead to more searching time, while large values may cause the optimal solution to be skipped. In this experiment, hyper-parameters of the algorithm are empirically determined as $\theta_1 = 2$, $\theta_2 = 3$, and $\lambda = 5$ in Experiment A and $\lambda = 3$ in Experiment B via several trials to find a balance between the optimization speed and modeling accuracy. The sigmoid function is considered as an activation function in all DNNs.

In DETL, the raw dataset is organized as shown in Fig. 5. The first 25% of raw data from each WT is extracted to form the source domain. In the source domain, 80% of data are randomly selected to develop a training set and the remaining 20% become the validation set. The next 30% of raw data from each WT are not utilized to show advantages of DETL in terms of requiring less training time and data in the model development. The following 30% of each raw dataset form target domains and the last 15% of raw data form test sets.

To compare prediction performances, a metric, the mean absolute percentage error (MAPE), expressed in (4) is employed. The MAPE depicts predictions errors in terms of the percentage, which directly reflects model accuracies.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4}$$

In (4), n describes the number of observations while y_i and $\hat{y}_i$ are the observed and predicted value of the i-th record.

4.4. Implementation details of the benchmarking methods

In IRDNN, each DNN have three hidden layers with 150, 100, 50 hidden units respectively in Experiment A while have all three hidden layers with 100 hidden units in Experiment B. In IAEDNN, to facilitate fair comparative analyses between DETL models and benchmarks, benchmarking DNNs have the same structure as that determined by DETL. Each WT dataset is split into training and test datasets by 85% and 15% for the benchmarking methods. Same test sets considered in DETL are utilized for testing.

4.5. Computational studies and results in experiment A

To develop prediction models with DETL, an AE model is first developed in source domain learning. Based on datasets mentioned in Section 4.3, network structures tried by DETL for AE models are reported in Table 4. The selected network structure is underlined. The maximal number of layers is controlled by a stopping criterion, $\theta_1 = 2$.

Results of Table 4 clearly present that the RE is generally increased by introducing extra hidden layers, and suggest that the AE with a single hidden layer developed via the source domain learning is selected as the base model for transferring to customized AE models through the target domain learning. It is worth noting that although the selected AE model contains only one hidden layer, the DETL framework is regarded as a deep learning framework due to its ability of generating DNNs.

Next, the base AE is fine-tuned with target domains and transformed into DNN prediction models. Mean values of predictions for 15 WTs are reported in Table 5, compared with results of benchmarking methods.

Computational results in Table 5 show that models developed by the DETL can yield a higher prediction accuracy for 15 WTs on average among all considered methods. IAEDNN could reach a similar prediction result compared to DETL; however, it requires much longer training time because each prediction model is individually trained from a random network parameter initialization. In IRDNN, even though the ReLU activation function could help accelerate the BP optimization and a deep architecture could increase the ability of learning complex functions, separately training models still costs a long time due to a deeper architecture and leads to a relatively lower prediction accuracy. Despite that the polynomial fitting in PM is computationally efficient, its large value of MAPE for one-step ahead predictions in Table 5 proves that training PM model is practically difficult. The physics-based model which is explicitly formulated as mathematical equations is based on ideal

Table 5
Average MAPE of 15 WTs of multi-step predictions in Experiment A.

Method	1-step	2-step	3-step	4-step	5-step	6-step	Average	Training time (s)
DETL	**8.42%**	**11.62%**	**13.67%**	**15.13%**	**16.33%**	**17.42%**	**13.77%**	40711
IRDNN	9.35%	12.50%	14.50%	15.91%	16.99%	18.06%	14.55%	49050
IAEDNN	8.68%	12.11%	14.20%	15.67%	16.79%	17.90%	14.22%	55492
PM	2211.81%	–	–	–	–	–	–	640
UDETL	13.89%	16.62%	18.82%	19.21%	21.24%	21.27%	18.51%	45637

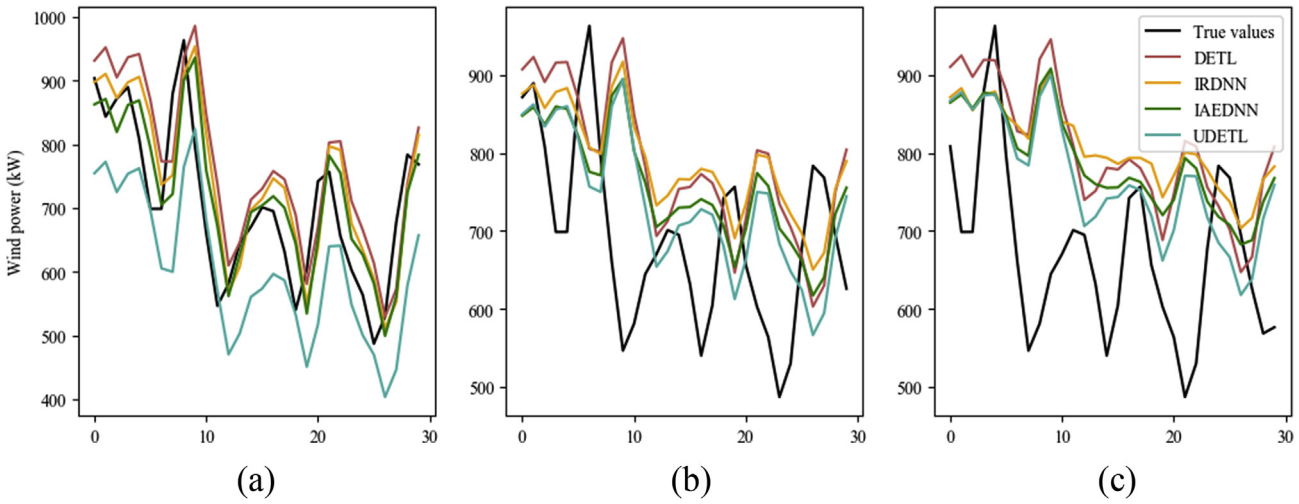


Fig. 9. Results of (a) 10-s, (b) 30-s, and (c) 50-s ahead predictions for a randomly selected WT.

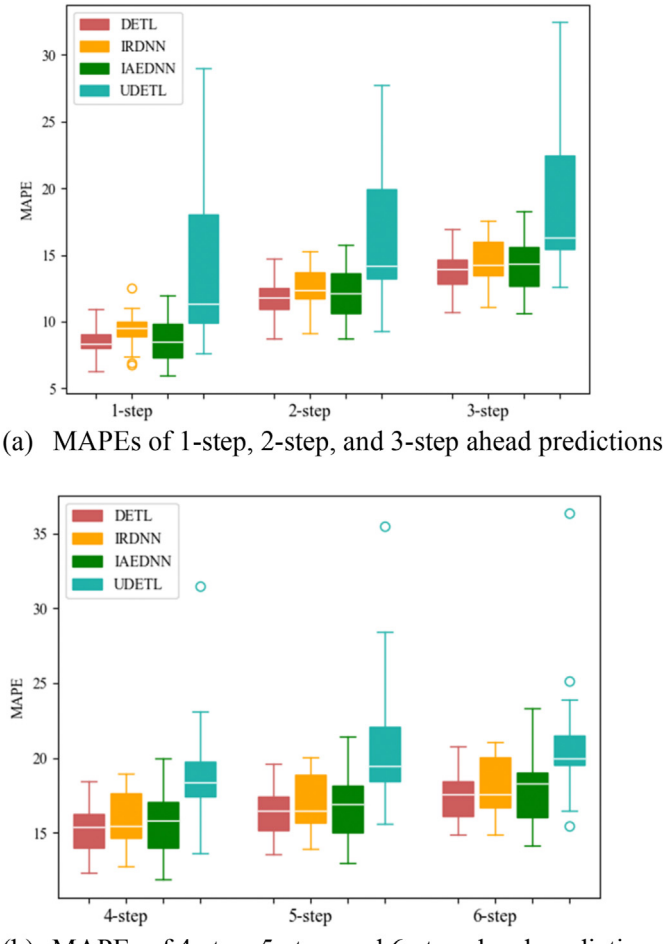


Fig. 10. Box-plots of errors for 15 WTs from 1-step ahead to 6-step ahead prediction in Experiment A.

assumptions for reality and ignores other possible influencing variables and vulnerable to the nonignorable noises. Finally, to train a unified prediction model for all WTs in UDETL, a huge number of training data are utilized and each epoch is significantly longer.

Table 6
REs of the DETL in source domain learning in Experiment B.

AE depth	AE structure	RE on validation set (× 0.001)
1	**22 - 57 - 22**	**0.702**
2	22 - 57–62 - 22	3.171
3	22 - 57–62 - 50 - 22	3.859

However, the unsatisfactory performance of UDETL indicates that a unified but compromised prediction model for a group of WTs is unrealistic. Prediction results of 1-, 3-, and 5-step ahead predictions conducted for a randomly selected WT, as shown in Fig. 9, could manifest results of Table 5.

In Fig. 10, to show the variation of prediction accuracy on 15 WTs, box-plots of prediction errors are presented. As the result of PM model significantly deviates from the other methods, for a better visualization, only the MAPEs of DETL, IRDNN, IAEDNN, and UDETL are involved in Fig. 10.

In Fig. 10, it is observable that considering median values, the prediction models trained by the DETL yield the lowest prediction errors for 15 WTs, which is similar to the conclusion obtained in Table 5. In addition, prediction errors on 15 WTs present the lowest variation in DETL as well, corresponding to the compact boxes and the shortest whiskers. The low variation reflects the robustness of prediction models, which is realized by a source domain learning to capture the homogeneous pattern and allows each prediction model to be developed based on the base model. In contrast, models developed by the UDETL give the largest variation of prediction errors, which means that a unified but compromised model would perform very differently in different WTs. Therefore, models trained by DETL can perform accurate wind power output predictions on average, and the predictions are more robust to avoid large biased results.

4.6. Computational studies and results in experiment B

As similar as in Experiment A, an AE base model is first developed via source domain learning. Network structures during the search process are reported in Table 6. The selected network structure is underlined.

Based on results of Table 6, the lowest RE on the validation set is obtained when the base AE has only one hidden layer. Compared with Table 4, REs in Experiment B are larger than those in

Table 7
Average MAPE of 50 WTs of multi-step predictions in the Experiment B.

Method	1-step	2-step	3-step	4-step	5-step	6-step	Average	Training time (s)
DETL	**27.25%**	**30.38%**	**32.63%**	**34.20%**	**35.64%**	**37.10%**	**32.86%**	14398
IRDNN	27.73%	30.80%	33.10%	34.73%	36.26%	37.91%	33.42%	19284
IAEDNN	32.27%	34.51%	36.16%	37.37%	38.70%	39.97%	36.50%	20751
UDETL	27.59%	30.97%	33.25%	34.94%	36.53%	38.31%	33.60%	16983

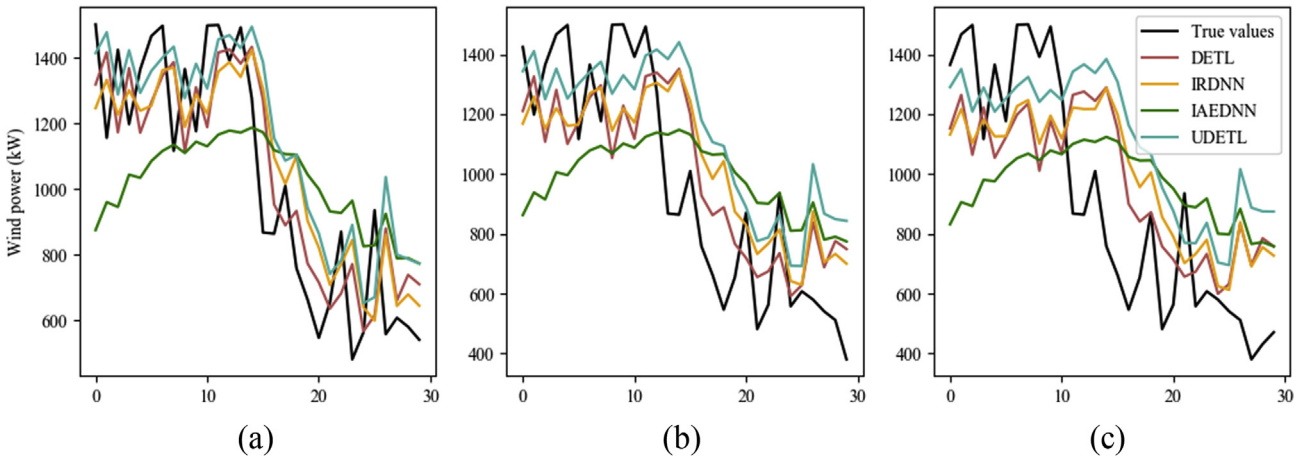


Fig. 11. Results of (a) 10-min, (b) 30-min, and (c) 50-min ahead predictions for a randomly selected WT.

Experiment A, which may imply that homogeneous patterns in Experiment B become less significant than those in Experiment A. Next, the base AE model is fine-tuned with target domain data and transformed into prediction models. MAPEs as prediction results are reported in Table 7.

In Table 7, all 50 prediction models developed by the DETL achieve the lowest prediction errors in each single step predictions and overall predictions. The accuracy of the IAEDNN is significantly lower than other methods. It could be probably due to the shortage of training data caused by a larger sampling interval, 10-min. In contrast, the gap between UDETL and DETL is reduced by comparing with results of Experiment A because the transfer learning approach requires less data for model training in both methods. Although the accuracy of the DETL is improved slightly by comparing with IRDNN and UDETL in the 1-step ahead prediction, its training efficiency is significantly enhanced according to the training time. It is worth noting that the training time is generally shorter than in Experiment A because the original data with a sampling interval of 10-s in the wind farm A were down-sampled for conducting 10-min ahead predictions. Such arrangement leads to less training data utilized in Experiment B overall. To compare and visualize prediction results, Fig. 11 shows 10-min-, 30-min- and 50-min ahead predictions of a randomly selected WT. It is observable that the wind power itself varies more irregularly than in Fig. 9, which would overall result in larger MAPEs.

Next, box-plots are provided in Fig. 12 to compare the accuracy variation of the batch of models.

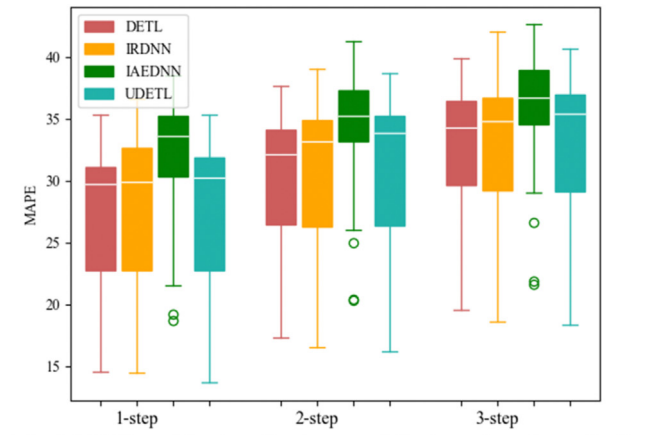
As shown in Fig. 12, models developed by DETL obtained the lowest median prediction accuracy among all methods and such result is consistent to results of Table 7. In addition, compact boxes and the shortest whiskers of DETL demonstrate the lowest variation as well, which proves the robustness of proposed method.

4.7. Summary of experimental results

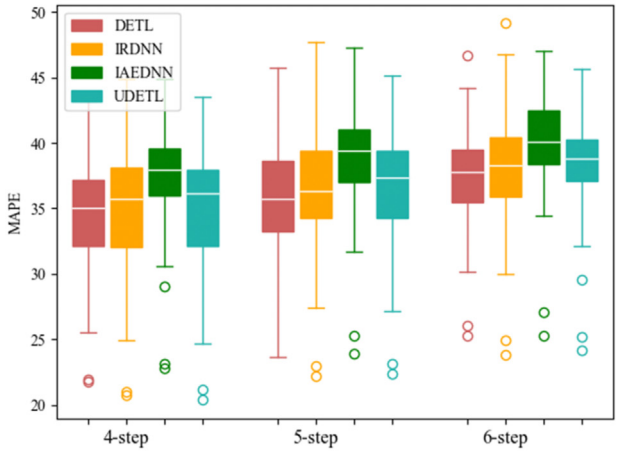
Based on presented computational studies, we can draw following insights:

a) The DETL could develop wind power prediction models for a group of WTs to yield a high prediction accuracy.
b) As the group of prediction models are evolved from the same base model, the performance of models trained by DETL is robust to present a more consistent prediction accuracy.
c) The training time of DETL is significantly reduced compared with conventional modeling methods due to two advantages. First, the determination of network architecture is optimized by adding new elements and updating network parameters rather than performing the conventional trials-and-errors, which completely randomly initializes and trains each candidate network. In addition, each prediction model can be fast evolved from a well trained base model which attempts to extract homogeneous patterns of WT system dynamics.
d) The DETL framework overall needs less training data, which could accelerate training when a huge amount of data are available, or save data when data are scarce.
e) DETL can adaptively determine an appropriate network structure. This can avoid the performance degradation caused by arbitrarily selecting a deep NN that may not help improve the modeling accuracy. In addition, exhaustive trials-and-errors for determining NN structures can be avoided.

In general, results of computational experiments demonstrate the superiority of DETL from a number of aspects. First, the DETL can well model homogeneous and heterogeneous characteristics of WTs to develop more accurate DNN prediction models with less data and shorter training time than the classical data-driven modeling framework. Results also indicate the significant impact of the DNN structure on modeling accuracy and validate advantages of automatically determining high quality NN structures. Moreover, the better performance of DETL also demonstrates



(a) MAPEs of 1-step, 2-step, and 3-step ahead predictions



(b) MAPEs of 4-step, 5-step, and 6-step ahead predictions

Fig. 12. Box-plots of errors for 50 WTs from 1-step ahead to 6-step ahead prediction in Experiment B.

significant benefits offered by transfer learning and indicates the existence of heterogeneous patterns among WTs.

Limitations of DETL are majorly related to stopping criteria of optimizing network structures, which check the continuous increase of validation errors. First, the greedy strategy might not guarantee a global optimum. In addition, checking the continuous increase of validation errors needs to over-expand network structures for certain iterations, which results in some additional computational costs.

5. Conclusion

In this paper, a novel deep and transfer learning framework, DETL, was proposed for batch-developing wind power output prediction models for a population of WTs in a wind farm from a data-driven perspective. The DETL applied three steps to first extract homogeneous patterns of system dynamics among a group of WTs and next model heterogeneous and temporal characteristics for each WT. In the first step, a self-organizing mechanism for automatically determining the structure of a base AE model was introduced to describe homogeneous patterns of considered WTs. Next, the transfer learning was applied by the DETL to convert the base AE to customized AE models for specifying heterogeneity in system dynamics among WTs. Finally, the customized AE models were transformed to DNN prediction models by incorporating temporal features and target variables onto the input layer and output layer, respectively. The SCADA data collected from 50 WTs in two commercial wind farms were utilized to study the proposed DETL. The implementation of DETL required a special arrangement of datasets, which organized a group of datasets into a source domain and multiple target domains. The source domain served the development of a base AE with DETL while target domains were utilized to produce customized AE models and prediction models for individual WTs based on the base AE. Through DETL, DNN models were developed to predict wind power outputs for each of considered WTs.

Computational experiments were conducted to validate the modeling accuracy and its stability of the proposed DETL. A couple of DNN prediction models developed without self-organizing and transfer learning processes, a unified DNN prediction model, and a physics-based model were considered as benchmarks. Two sets of experiments considering different prediction horizons as well as different number of WTs and SCADA parameters were separately performed. Computational results verified the advantage of DETL in determining decent structures of DNNs as well as offering a higher prediction accuracy and efficiency by comparing with benchmarks.

In future studies, we plan to extend the DETL to study more application problems of more large-scale distributed systems to study the practical value of the proposed DETL in other problems.

Author contribution

Xin Liu: Methodology; Validation; Formal analysis; Writing – original draft, Zheming Cao: Investigation, Zijun Zhang: Conceptualization; Supervision; Writing – review & editing; Funding acquisition

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.

Acknowledgements

This research was supported in part by the Hong Kong Research Grants Council General Research Fund with No. 11215418, in part by the Project under National Natural Science Foundation of China for the Youth Scientist with No. 52007160, and in part by the CityU Strategic Research Grant with No. 7005302.

References

[1] Giorgi MGD, Ficarella A, Tarantino M. Assessment of the benefits of numerical weather predictions in wind power forecasting based on statistical methods. Energy 2011;36(7):3968–78.
[2] Khalid M, Savkin AV. A method for short-term wind power prediction with multiple observation points. IEEE Trans Power Syst 2012;27(2):579–86.
[3] Stathopoulos C, Kaperoni A, Galanis G, Kallos G. Wind power prediction based on numerical and statistical models. J Wind Eng Ind Aerod 2013;112:25–38.
[4] Kusiak A, Zheng H, Song Z. Short-term prediction of wind farm power: a data mining approach. IEEE Trans Energy Convers 2009;24(1):125–36.
[5] Bao Y, Xiong T, Hu Z. Multi-step-ahead time series prediction using multiple-output support vector regression. Neurocomputing 2014;129:482–93.
[6] Croonenbroeck C, Ambach D. A selection of time series models for short- to medium-term wind power forecasting. J Wind Eng Ind Aerod 2015;136:201–10.
[7] Nielson J, Bhaganagar K, Meka R, Alaeddini A. Using atmospheric inputs for Artificial Neural Networks to improve wind turbine power prediction. Energy 2020;190(116273).
[8] Heinermann J, Kramer O. Machine learning ensembles for wind power prediction. Renew Sustain Energy Rev 2016;89:671–9.
[9] Yuan X, Tan Q, Lei X, Yuan Y, Wu X. Wind power prediction using hybrid autoregressive fractionally integrated moving average and least square support vector machine. Energy 2017;129:122–37.

[10] Dong L, Wang L, Khahro SF, Gao S, Liao X. Wind power day-ahead prediction with cluster analysis of NWP. Renew Sustain Energy Rev 2016;60:1206–12.
[11] Wang H, Li G, Wang G, Peng J, Jiang H, Liu Y. Deep learning based ensemble approach for probabilistic wind power forecasting. Appl Energy 2017;188:56–70.
[12] Hu Q, Zhang R, Zhou Y. Transfer learning for short-term wind speed prediction with deep neural networks. Renew Energy 2016;85:83–95.
[13] Yin H, Ou Z, Huang S, Meng A. A cascaded deep learning wind power prediction approach based on a two-layer of mode decomposition. Energy 2019;189(116316).
[14] Wu W. Neural network structure optimization based on improved genetic algorithm. In: 5th int. Conf. Adv. Comput. Intell. (ICACI), nanjing, China; 2012. p. 893–5.
[15] Chang CC, Chang TYP, Xu YG. Adaptive neural networks for model updating of structures. Smart Mater Struct 2000;9:59–68.
[16] Abedinia O, Amjady N. Short-term wind power prediction based on hybrid neural network and chaotic shark smell optimization. Int J Precis Eng Manuf 2015;2(3):245–54.
[17] Qureshi AS, Khan A, Zameer A, Usman A. Wind power prediction using deep neural network based meta regression and transfer learning. Appl Soft Comput 2017;58:742–55.
[18] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng 2010;22(10):1345–59.
[19] Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzz. 1998;6(2):107–16.
[20] Hinton G, Salakhutdinov R. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7.
[21] Bengio Y, Lamblin P, Popovici D, Larochelle H. Greedy layer-wise training of deep networks. in *Proc. Adv. Neural Inf. Process. Syst.* 2007:153–60.
[22] Khodayar M, Kaynak O, Khodayar ME. Rough deep neural architecture for short-term wind speed forecasting. IEEE Trans. Ind. Informat. Dec. 2017;13(6):2770–9.
[23] Jiao R, Huang X, Ma X, Han L, Tian W. A model combining stacked auto encoder and back propagation algorithm for short-term wind power forecasting. IEEE Access 2018;6:17851–8.
[24] Jiang G, He H, Xie P, Tang Y. Stacked multilevel-denoising autoencoders: a new representation learning approach for wind turbine gearbox fault diagnosis. IEEE Trans. Instrum. Meas. Sept. 2017;66(9):2391–402.
[25] Ma M, Sun C, Chen X. Deep coupling autoencoder for fault diagnosis with multimodal sensory data. IEEE Trans. Ind. Informat. March 2018;14(3):1137–45.
[26] Wang L, Zhang Z, Xu J, Liu R. Wind turbine blade breakage monitoring with deep autoencoders. IEEE Trans. Smart Grid 2018;9(4):2824–33.
[27] Sakurada M, Yairi T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: 2nd workshop on machine learning for sensory data analysis (MLSDA). Australia: Gold Coast; 2014.
[28] Saufi SR, Ahmad ZAB, Leong MS, Lim MH. Gearbox fault diagnosis using a deep learning model with limited data sample. IEEE Trans. Ind. Informat Oct. 2020;16(10):6263–71.
[29] Gondara L, Wang K. MIDA: multiple imputation using denoising autoencoder. In: Pacific-asia conference on knowledge discovery and data mining. Springer; 2018. p. 260–72.
[30] Hong Y-Y, Rioflorido CLPP. A hybrid deep learning-based neural network for 24-h ahead wind power forecasting. Appl Energy 2019;250:530–9.
[31] Wang L, Zhang Z, Long H, Xu J, Liu R. Wind turbine gearbox failure identification with deep neural networks. IEEE Trans. Ind. Informat. 2017;13(3):1360–8.
[32] Díaz–Vico D, Torres–Barrán A, Omari A, Dorronsoro JR. Deep neural networks for wind energy prediction. Neural Process Lett 2017;46(3):829–44.
[33] Jolliffe IT. Principal component analysis. New York: Springer-Verlag; 1986.
[34] Xavier G, Bordes A, Bengio Y. Deep sparse rectifier neural networks. in *Proc. 14th Int. Conf. Artif. Intell. Stat.* 2011.
[35] Boukhezzar B, Siguerdidjane H. Nonlinear control of variable speed wind turbines without wind speed measurement. in *Proc. 44th IEEE Conf. Decis. Control* 2005:3456–61.