Renewable Energy 85 (2016) 83–95

Contents lists available at ScienceDirect

Renewable Energy

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

Transfer learning for short-term wind speed prediction with deep neural networks

Qinghua Hu*, Rujia Zhang, Yucan Zhou

School of Computer Science and Technology, Tianjin University, Tianjin, China

A R T I C L E I N F O

Article history:
Received 19 December 2014
Received in revised form
6 April 2015
Accepted 15 June 2015
Available online 25 June 2015

Keywords:
Wind speed prediction
Transfer learning
Deep neural networks
Stacked denoising autoencoder

A B S T R A C T

As a type of clean and renewable energy source, wind power is widely used. However, owing to the uncertainty of wind speed, it is essential to build an accurate forecasting model for large-scale wind power penetration. Numerical weather prediction (NWP) and data-driven modeling are two typical paradigms. NWP is usually unavailable or spatially insufficient. Data-driven modeling is an effective candidate. As to some newly-built wind farms, sufficient historical data is not available for training an accurate model, while some older wind farms may have long-term wind speed records. A question arises regarding whether the prediction model trained by data coming from older farms is also effective for a newly-built farm. In this paper, we propose an interesting trial of transferring the information obtained from data-rich farms to a newly-built farm. It is well known that deep learning can extract a high-level representation of raw data. We introduce deep neural networks, trained by data from data-rich farms, to extract wind speed patterns, and then finely tune the mapping with data coming from newly-built farms. In this way, the trained network transfers information from one farm to another. The experimental results show that prediction errors are significantly reduced using the proposed technique.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Owing to the depletion of conventional energy sources and the deterioration of the environment, clean and renewable energy sources are being widely utilized all around the world. With the advantages of non-pollution, low costs and remarkable benefits of scale, wind power is considered as one of the most important sources of energy [1]. However, wind energy is uncertain and variable. If the wind penetration power exceeds a certain value, the quality of the generated power and the power system may both be seriously affected [2]. As the most important factor of wind power, wind speed needs to be predicted accurately [3]. Accurate prediction of wind speed is important for the allocation, scheduling, maintenance, and planning of wind energy conversion systems [4–6].

Different kinds of methods have been proposed to handle wind speed forecasting. These methods can be classified into four categories: i) physical models, ii) conventional statistical models, iii) spatial correlation models, and iv) artificial intelligence and new models [7]. A physical model uses physical or meteorology information such as the temperature, pressure and orography to predict the future speed [8]. It has advantages in long-term prediction, but it does not give accurate results for short-term prediction [9]. A conventional statistical model uses historical wind speed data for training. The goal is to find the relationship between certain explanatory variables and future wind speed. The representative statistical models are autoregressive model (AR), moving average model (MA), autoregressive moving average model (ARMA), and autoregressive integrated moving average model (ARIMA) [10]. A spatial correlation model takes into account the spatial relationship of the wind speed in different wind speed measurement stations. The wind speed time series of the predicted points and its neighboring observation points are employed to predict the wind speed [2,11]. This model is more difficult to develop than typical time series models. In addition, with the development of artificial intelligence and other forecasting methods, various new models have been proposed, such as artificial neural networks (ANNs) [12–16], support vector regression machines (SVR) [17–20], and various hybrid methods [21–24].

There are many wind farms located in different areas. For wind speed prediction, different models can be learned for different farms. This is feasible when the amount of data is sufficient.

* Corresponding author.
E-mail address: huqinghua@tju.edu.cn (Q. Hu).

http://dx.doi.org/10.1016/j.renene.2015.06.034
0960-1481/© 2015 Elsevier Ltd. All rights reserved.

However, for a newly-built wind farm, insufficient wind speed data is available for the model design. Togelou et al. proposed a solution to this problem by providing two statistical models that are self-constructed and self-adapted online [25]. They did not consider data from any other sites. An alternative strategy is to learn a model by mixed data from the target domain and other domains together. This makes sense only if the data from other farms can be directly used to train the target model. Since wind speed patterns are different across domains, the above solution is usually infeasible. Therefore, another strategy has been proposed [26]. This strategy refers to the learning of a shared model from the set of domains, and then adapting the model to each individual domain. This only works, however, when the model is able to discover intermediate abstractions that are shared and useful across domains.

In recent years, there has been considerable interests in developing multilingual speech technologies. Most of these works focus on transferring knowledge between languages because of the small amount of resources for the target language. The improvement is based on the discovery of the commonalities between different languages through deep learning [27–31]. Most works combine multiple languages. Among them, some are trained in a sequential way [28]. Several others adopt a parallel training strategy in which all languages are trained simultaneously [29–31]. Such works on multilingual technologies are amazing. They motivated our proposal of a deep learning approach for the problem of short-term wind speed forecasting. We fuse the wind speed information from multi-sources to build a DNN. The hidden layers are shared across many farms while the output layers are farm dependent. The shared hidden layers can be considered as a universal feature transformation that works well for many domains.

This paper offers the following three contributions:

- We propose a deep architecture to extract the hidden rules of wind speed patterns. To the best of our knowledge, this paper is the first to use deep learning for wind speed prediction.
- A new research question for wind speed prediction is introduced, i.e., transferring knowledge from data-rich farms to a newly-built farm. To the best of our knowledge, this is the first time transfer learning has been used in such an application.
- A series of experiments were conducted for this paper. We achieved an effective strategy for transfer learning. The experimental results show that the prediction errors are significantly reduced using the proposed technique.

The rest of this paper is organized as follows. In Section 2, we analyze the wind speed data-sets used for this study. In Section 3, we describe our proposed shared-hidden-layer DNN and some background knowledge. The prediction results for short-term wind speeds are presented and analyzed in Section 4. Finally, we offer some concluding remarks regarding this research in Section 5.

2. Statistical analysis of data

We collect some historical records of wind speeds from four wind farms located in Ningxia, Jilin, Inner Mongolia and Gansu, China. Their locations are shown in Fig. 1. Gansu is far away from Jilin. The distance between them is about 2800 km, while the farms located in Gansu and Ningxia are the nearest, and it is about 300 km. All these farms are located in the north of China, and mainly belong to the temperate zone of continental monsoon climate. Owing to the differences of weather, terrain, topography and so on, the characteristics of wind speeds in different wind farms are distinct. And strong wind occurs in different seasons and different times of the day. The data from these four farms are all sampled every 5 s. However, in our databases, the data are recorded every 10 min. Each data point is the average in a 10-min span. The length of time series available at Ningxia farm is 1.5 years, and the lengths of other three farms are 1 year. One-year wind speed data from the four wind farms are shown in Fig. 2. The annual wind speeds of four sites are around 5.5, 6.0, 7.5 and 7.5 m/s, respectively.

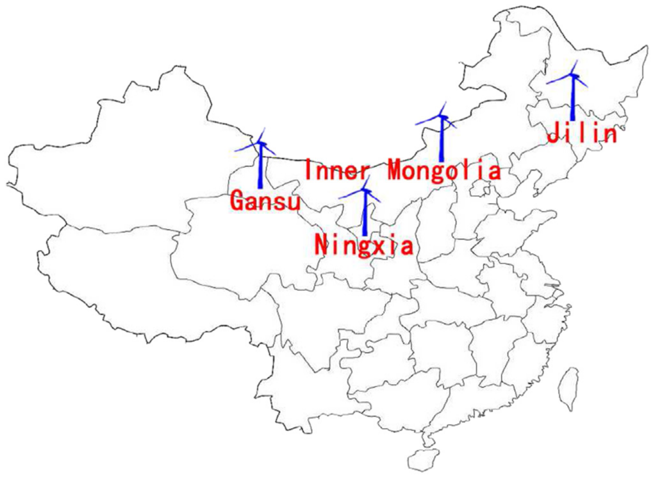

Fig. 1. Position of wind farms where we collect data of wind speed. The four wind farms are located in Ningxia, Jilin, Inner Mongolia and Gansu, respectively.

Fig. 3 shows the 1-h wind speed probability density histograms and the corresponding fitting functions. We can see that the Gaussian distribution is well fitted to the wind speed in a short time period. Fig. 4 shows 1-year wind speed probability density histograms and the Weibull distribution. From these figures, we see that the Weibull distribution is approximately fitted to the wind speed in 1-year period. This curve is widely accepted as a statistical description of wind speed [32]. The Weibull distribution is a unimodal continuous distribution with two parameters. Its probability density function is expressed as

$$f(x;\lambda,k)=\begin{cases}\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-x/\lambda^k} & x\geq 0\\ 0 & x<0\end{cases} \tag{1}$$

where x is a random variable, $\lambda>0$ is a scale parameter that determines the scale of the distribution and $k>0$ is a shape parameter that determines the shape of the distribution.

3. DNN-transfer models

Since the combination of deep learning and transfer learning is successfully applied in many tasks [26,31,33,34], we try to use it for wind speed prediction of a newly-built farm. This section describes our proposed shared-hidden-layer DNN model and some involved background knowledge.

3.1. *Deep learning approach*

The concept of deep learning originates from research on artificial neural networks. However, deep learning alleviates the local optima problem in the non-convex objective function of a deep network. Three techniques, namely, a large number of hidden units, better learning algorithm, and better parameter initialization technique, have contributed to the success of the deep learning approach [35]. In addition, deep architectures appear to be fit for representing higher-level abstractions [36]. For example, some of the features are useful across many domains, making deep learning

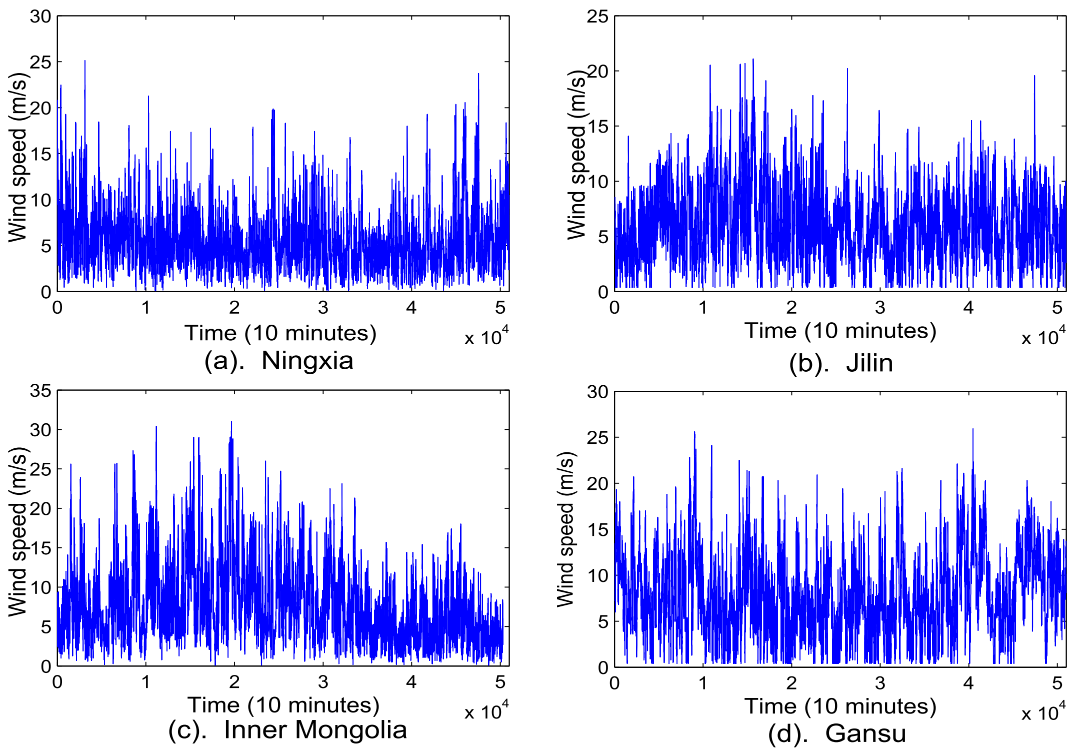

Fig. 2. One-year wind speed data from four wind farms.

particularly well-suited for transfer learning.

3.2. *Stacked denoising autoencoders*

The basic framework of our model is the stacked denoising autoencoder [37]. A deep autoencoder is a special type of DNN whose output has the same dimensions as the input. First, we will briefly describe the traditional autoencoder. An autoencoder is an unsupervised learning algorithm that sets the target values to be equal to the inputs. It is trained to encode the input $\mathbf{x} \in [0,1]^d$ into a

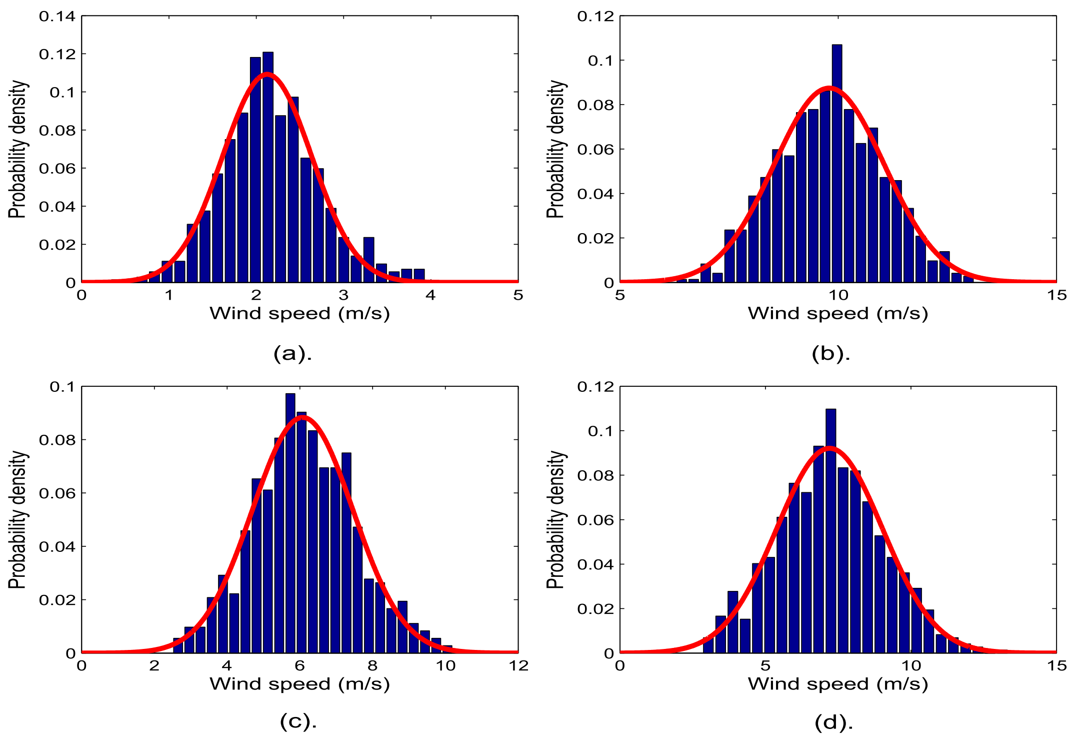

Fig. 3. Probability distributions of wind speed in an hour.

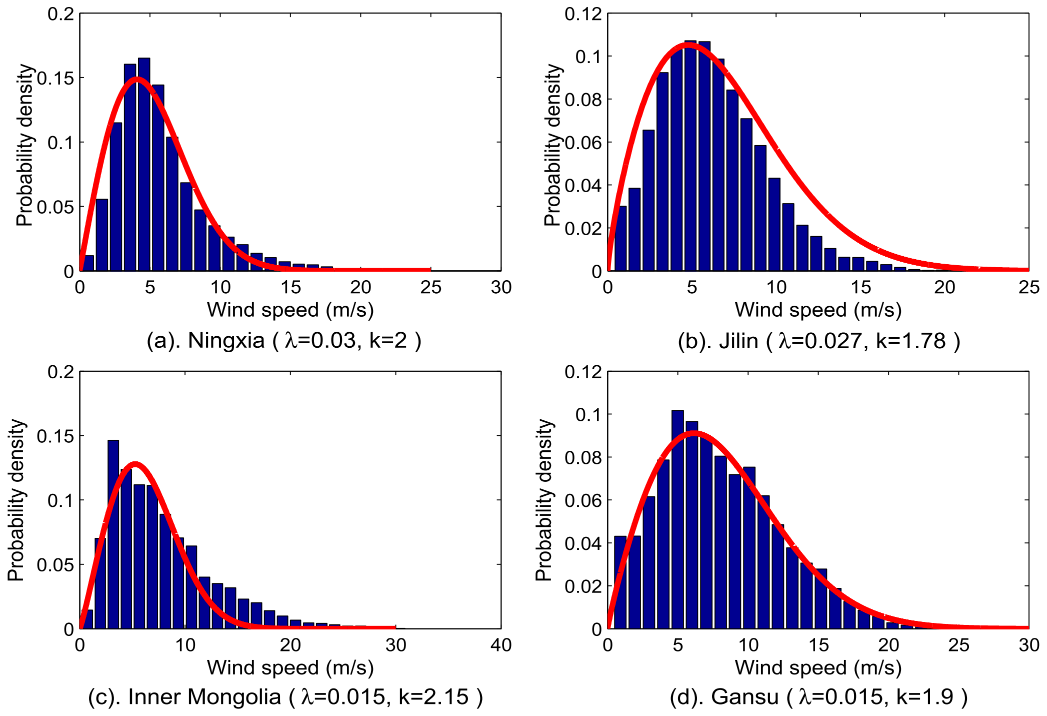

Fig. 4. Probability distributions of wind speed in a year. The values of Weibull's parameters of each data-set are recorded below each subfigure.

hidden representation $\mathbf{y}\in[0,1]^{d'}$ through a deterministic mapping:

$$\mathbf{y}=f_\theta(\mathbf{x})=s(\mathbf{W}\mathbf{x}+\mathbf{b}) \tag{2}$$

where s is a nonlinear mapping, such as the sigmoid function used in our experiments. Its parameter set is $\theta=\{\mathbf{W},\mathbf{b}\}$, where $\mathbf{W}$ is a $d'\times d$ weight matrix, and $\mathbf{b}$ is a bias vector of dimensionality d'. The latent representation, or code $\mathbf{y}$, is then decoded into a reconstruction of $\mathbf{z}\in[0,1]^d$ which has the same shape as $\mathbf{x}$ through a similar transformation:

$$\mathbf{z}=g_{\theta'}(\mathbf{y})=s(\mathbf{W}'\mathbf{y}+\mathbf{b}') \tag{3}$$

where $\mathbf{z}$ is seen as a prediction of $\mathbf{x}$, with $\theta'=\{\mathbf{W}',\mathbf{b}'\}$.The weight matrix $\mathbf{W}'$ of the reverse mapping may be optionally constrained by Refs. $\mathbf{W}'=\mathbf{W}^T$, which is an instance of tied weights. Each training $\mathbf{x}^{(i)}$ is thus mapped to a corresponding $\mathbf{y}^{(i)}$ and a reconstruction $\mathbf{z}^{(i)}$. The parameters of this model, namely θ and θ', are optimized to minimize the average reconstruction error:

$$\begin{aligned}\theta^*,\theta'^* &= \operatorname{argmin}_{\theta,\theta'}\frac{1}{n}\sum_{i=1}^{n}L(\mathbf{x}^{(i)},\mathbf{z}^{(i)})\\ &= \operatorname{argmin}_{\theta,\theta'}\frac{1}{n}\sum_{i=1}^{n}L\left(\mathbf{x}^{(i)},g_{\theta'}(f_\theta(\mathbf{x}^{(i)}))\right)\end{aligned} \tag{4}$$

where L is a loss function such as the traditional squared error $L(\mathbf{x},\mathbf{z})=\|\mathbf{x}-\mathbf{z}\|^2$.

In our models, we use a DAE instead of an ordinary autoencoder. The idea behind the DAE is simple. To force the hidden layer to discover more robust features and prevent it from simply learning the identity, the input is replaced by a partially destroyed version $\tilde{\mathbf{x}}$ by means of a stochastic mapping $\tilde{\mathbf{x}}\sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$. The input can be corrupted in many ways, and we use the original corruption mechanism which is parameterized by the desired proportion v of “destruction”. Particularly speaking, for each input $\mathbf{x}$, a fixed number vd of components are chosen at random, and their values are set to 0, while the others are left untouched [37]. The corrupted input $\tilde{\mathbf{x}}$ is then mapped, as with the basic autoencoder, to a hidden representation:

$$\tilde{\mathbf{y}}=f_\theta(\tilde{\mathbf{x}})=s(\mathbf{W}\tilde{\mathbf{x}}+\mathbf{b}) \tag{5}$$

and a reconstruction:

$$\tilde{\mathbf{z}}=g_{\theta'}(\tilde{\mathbf{y}})=s(\mathbf{W}'\tilde{\mathbf{y}}+\mathbf{b}') \tag{6}$$

By setting the latent representation (encoded output) of the DAE found on the layer below as an input to the current layer, the denoising autoencoder can be stacked to form a deep network. The unsupervised pre-training of such an architecture is conducted one

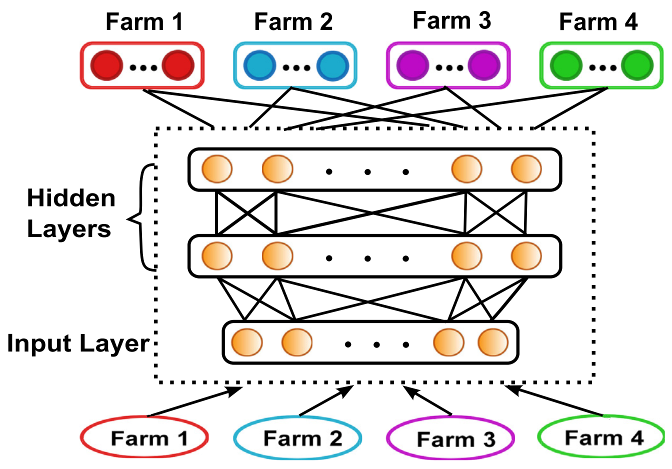

Fig. 5. Architecture of the shared-hidden-layer DNN.

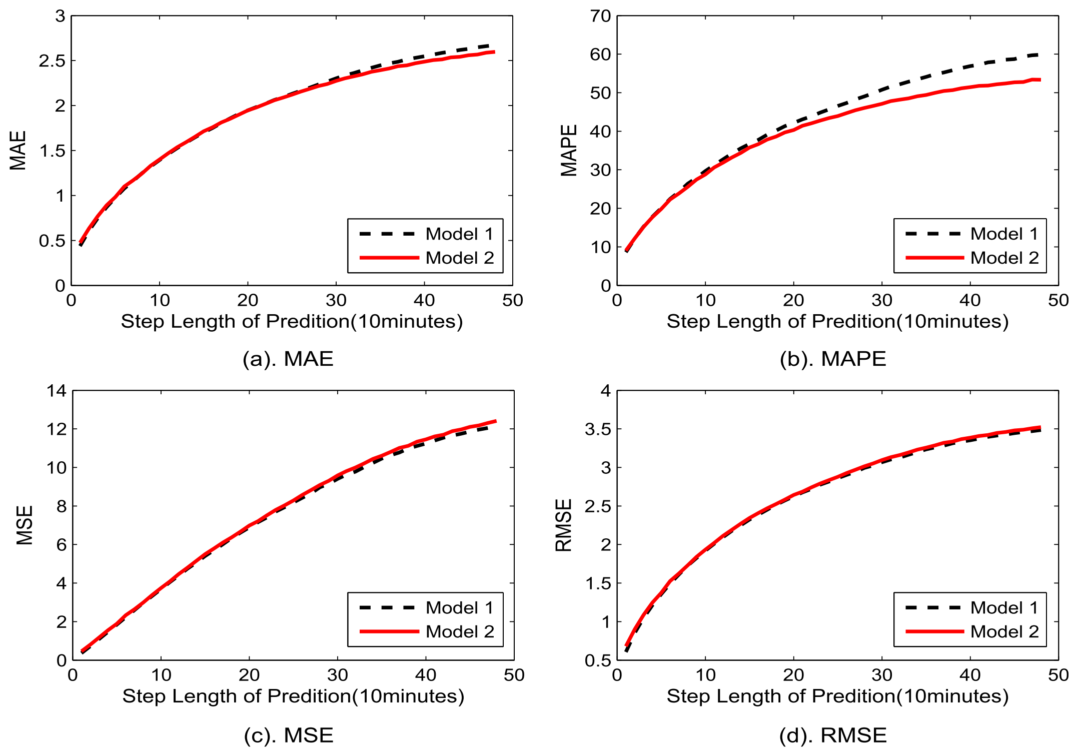

Fig. 6. Comparison of different prediction errors on Ningxia data whether trained with data from other farms together. Descriptions about each model are included in the text.

layer at a time. Once each layer is pre-trained, the parameters can be used as initialization for a network optimized with respect to the supervised training criterion [38], or a logistic regression layer can be directly added on top of the network.

3.3. Shared-hidden-layer DNN

Fig. 5 shows the architecture of the proposed shared-hidden-layer DNN (SHL-DNN), which is similar to the model used in

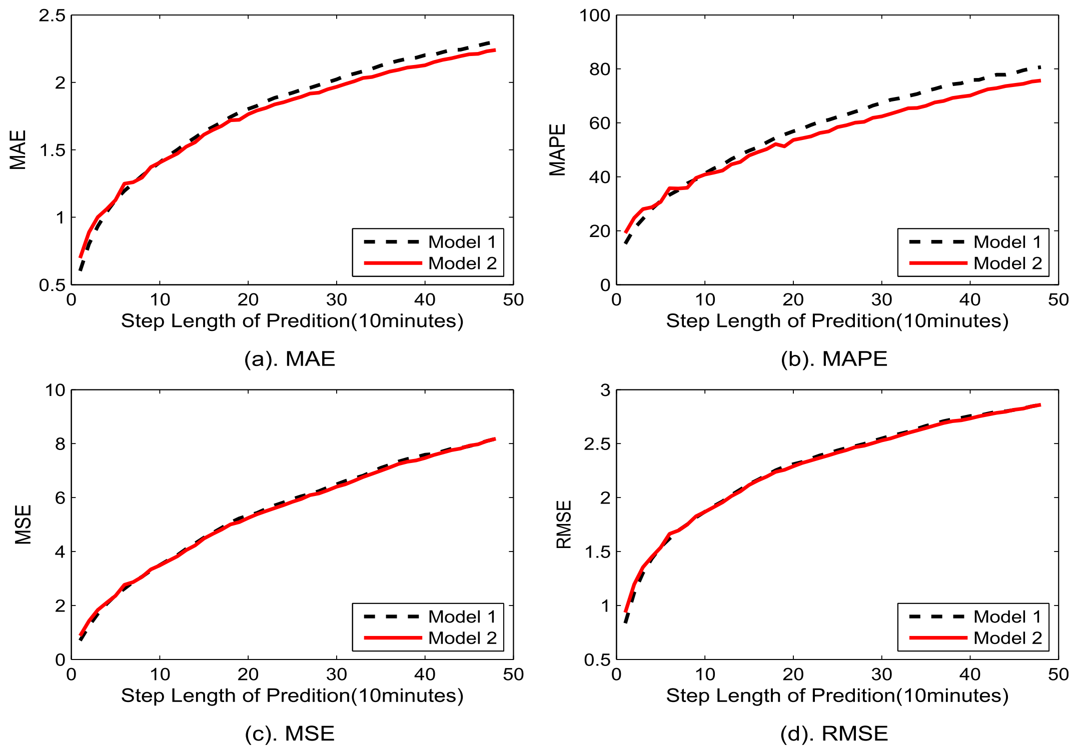

Fig. 7. Comparison of different prediction errors on Jilin data whether trained with data from other farms together. Descriptions about each model are included in the text.

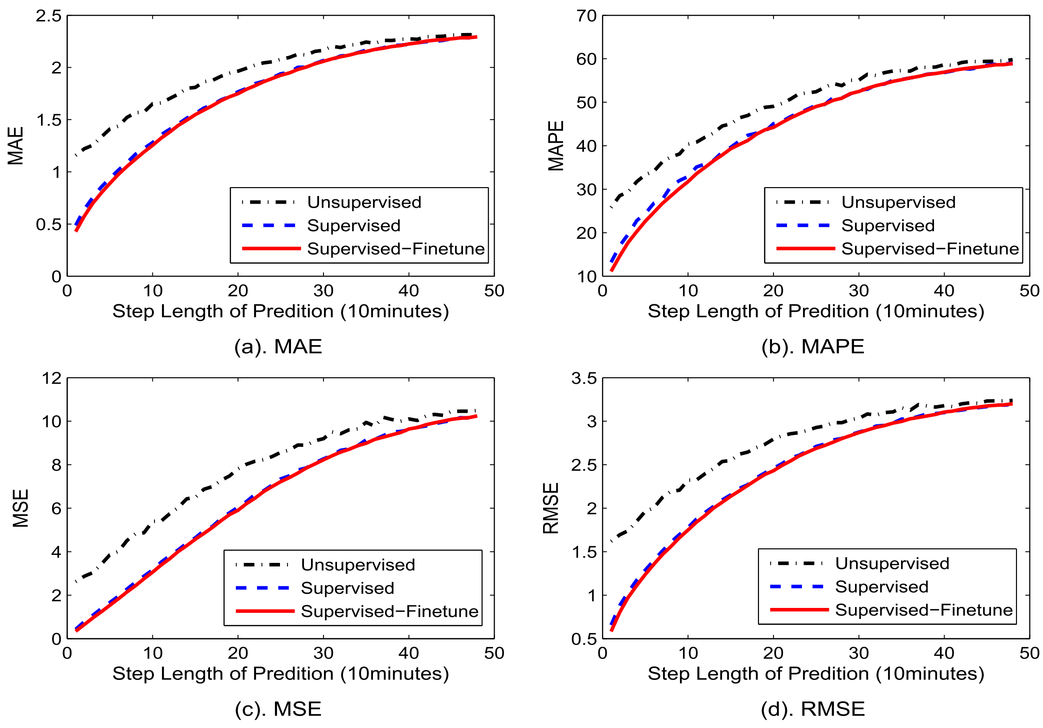

Fig. 8. Predicting performance of different training strategies on SHL-DNN model with respect to Ningxia data.

Ref. [38]. In this architecture, both the input and the hidden layers are shared across all wind farms and can be thought of as a common feature transformation. However, the output layers are not shared but remain separate from each other. Each farm has its own output layer since its data distribution differs from that of other farms. This is a type of knowledge transfer in which universal features are transferred to each data-set.

Huang et al. show that a parallel training strategy is better than a sequential training strategy [29]. That is the reason why we use such a shared-hidden-layer model. Three frequently-used strategies for training such a deep model are employed in our paper. The first training strategy, denoted as Unsupervised, uses only an

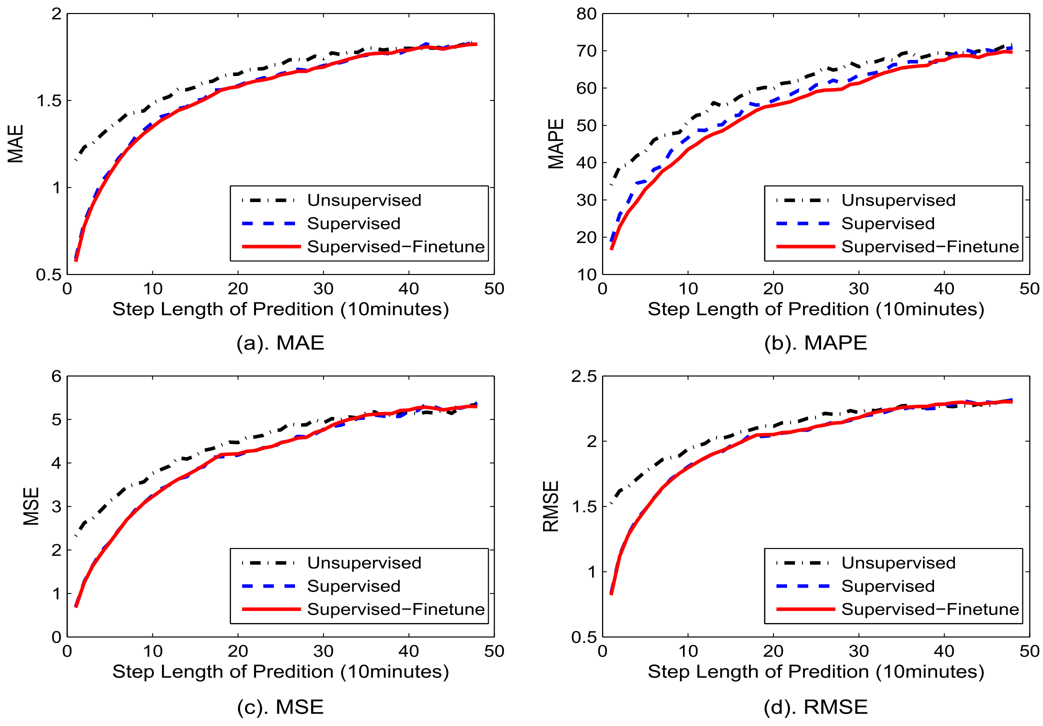

Fig. 9. Predicting performance of different training strategies on SHL-DNN model with respect to Jilin data.

Table 1
Different settings of three tests.

Settings	Target domain	Source domain
Test 1	Ningxia (0.5-month)	Jilin (1-year)
		Inner Mongolia (1-year)
		Gansu (1-year)
Test 2	Jilin (0.5-month)	Ningxia (1.5-year)
		Inner Mongolia (1-year)
		Gansu (1-year)
Test 3	Inner Mongolia (0.5-month)	Ningxia (1.5-year)
		Jilin (1-year)
		Gansu (1-year)

Note: The number in parenthesis is the size of the corresponding data-set.

unsupervised pre-training procedure (SDA) in the shared hidden layers. The second algorithm, denoted as Supervised, uses the label information for conventional back-propagation to fine tune the network after the pre-training. This is also only done for the hidden layers. The last method is similar to the second strategy, but involves not only hidden layers but also output layers. It is denoted by Supervised-Finetune. All three strategies described above are followed by training the output layer of each farm using its own labeled data. For the third strategy, it is the retraining or fine-tuning of the output layers. Thus, every model has four different outputs for the four wind farms.

4. Experiments

In this section, all of the experimental results for short-term wind speed prediction are presented. Real-world wind speed data-sets, namely Ningxia, Jilin, Inner Mongolia and Gansu, are used in this application. We totally perform four experiments. And different sized data-sets are employed for the different experiments.

The forecasting results were evaluated with four criteria, i.e., mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE) and mean absolute percentage error (MAPE). They are computed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i^{'}\right| \tag{7}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i^{'}\right)^2 \tag{8}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i^{'}\right)^2} \tag{9}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - y_i^{'}\right|}{y_i} \times 100\% \tag{10}$$

where n is the size of the sample, y_i is the ith real value and $y_i^{'}$ is the ith predicted value.

MAE evaluates the disparity between the real values and the predicted values. This function is more robust to the large errors than the other functions, while MSE and RMSE also reflect the dispersion of models. However, they are sensitive to the large errors compared with MAE because the errors are squared and the large errors are amplified further. MAPE is the ratio between errors and real values. It can be considered as a relative error function. A small error in a lower wind speed period may have large impact on this function, while a large error in a high wind speed may also have small effect on it. These four functions can be used to measure the performance of a forecasting algorithm from different viewpoints.

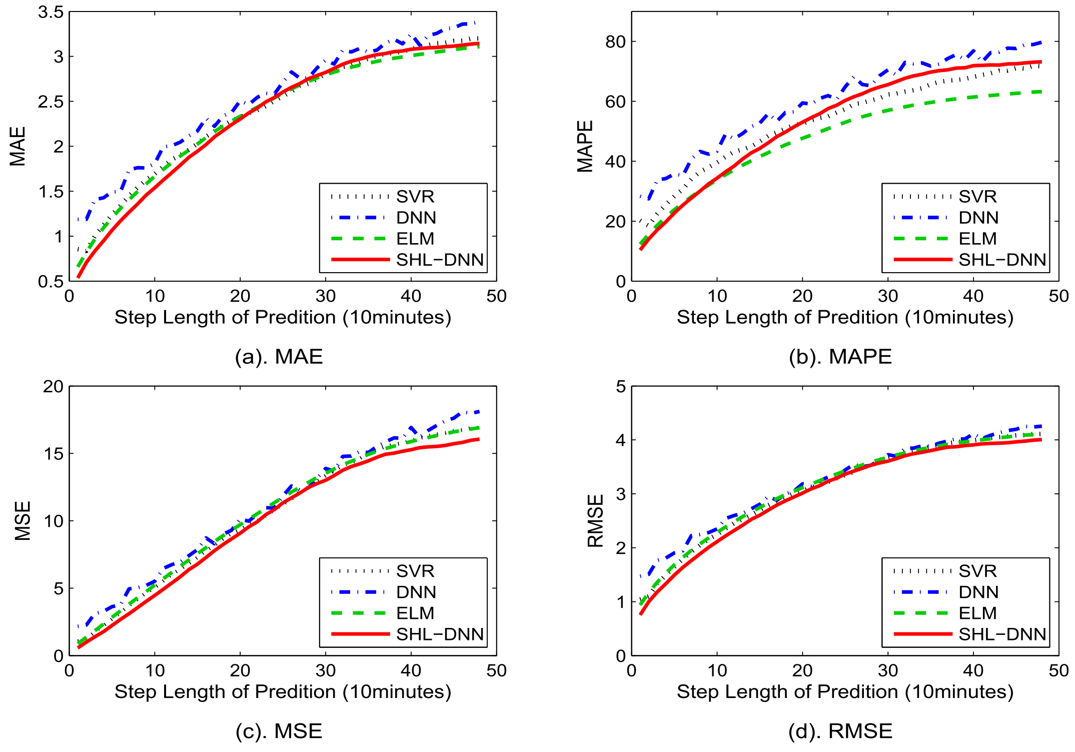

Fig. 10. Different prediction errors of three models on Ningxia data.

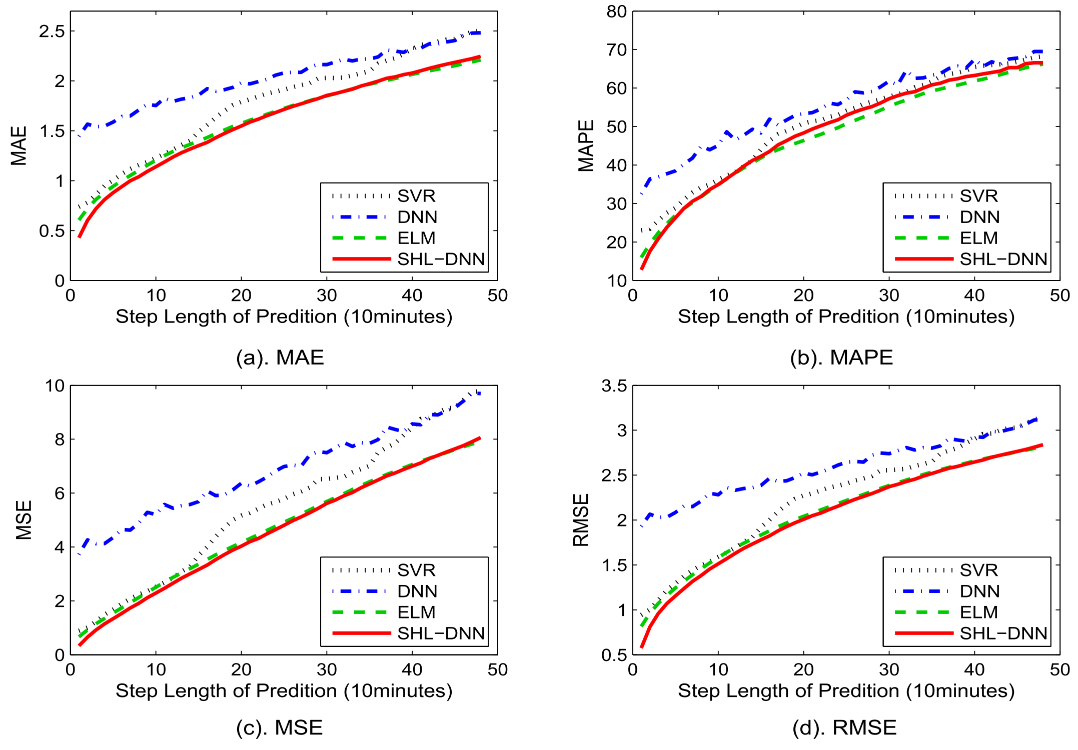

Fig. 11. Different prediction errors of three models on Jilin data.

4.1. Experimental setup

All experiments are performed on the model shown in Fig. 5. The deep model we used has two hidden layers, and the number of each hidden layer's nodes is 100. The destroyed fraction used for making the input zero is set to 0.5. We choose the sigmoid function as the activation function. The learning rates of the DAE and supervised neural network are 0.1 and 0.002, respectively.

All of the data-sets record the wind speed values in 10-min intervals. In our experiments, 1-day data values (144 speeds) are used as the input variables $x = (x_{i-143}, x_{i-142}, \ldots, x_{i-1}, x_i)$. The task is to predict the wind speeds $y = (x_{i+1}, x_{i+2}, \ldots, x_{i+48})$ for the next 8 h.

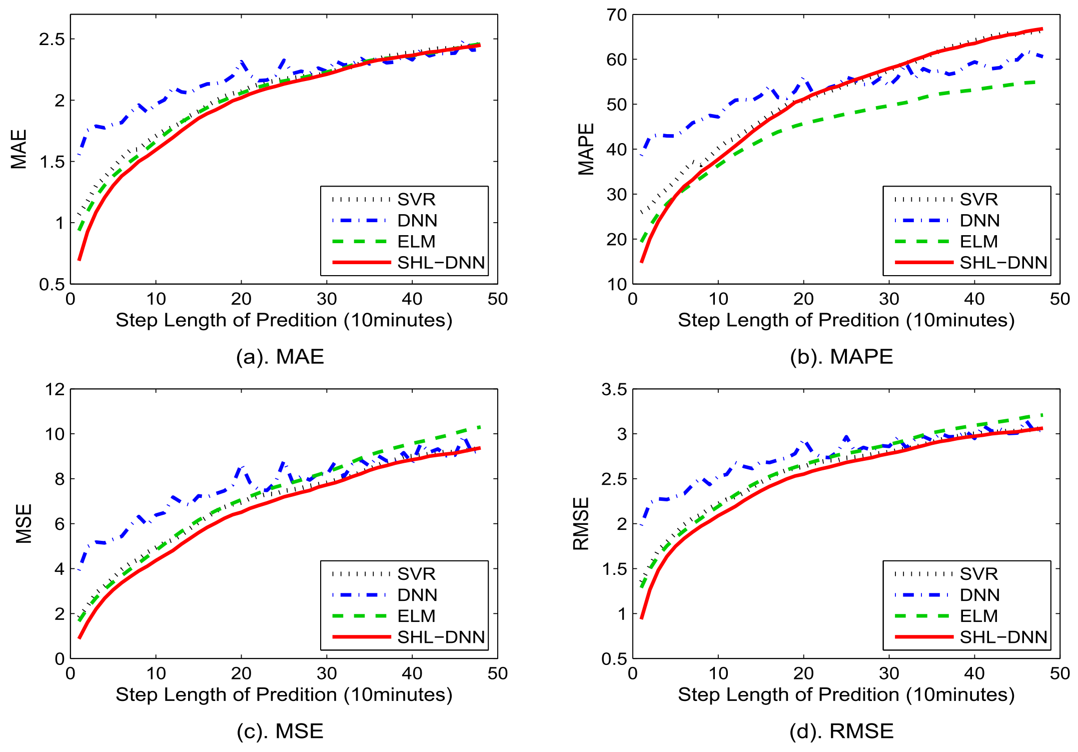

Fig. 12. Different prediction errors of three models on Inner Mongolia data.

Table 2
MAE of four models after different time scales on Ningxia data. The best performances are marked in boldface.

Time	Model	Training data of target domain						
		0.5-month	1-month	2-month	3-month	4-month	5-month	6-month
10-min	DNN	1.1865	0.8269	0.8361	0.6770	0.6025	0.6054	0.5861
	SVR	0.8545	0.7847	0.8022	0.8034	0.7630	0.7699	0.7670
	ELM	0.6578	0.6357	0.6311	0.6311	0.6295	0.6316	0.6269
	SHL-DNN	**0.5348**	**0.4927**	**0.4886**	**0.4882**	**0.4803**	**0.4879**	**0.4679**
30-min	DNN	1.4072	1.0586	1.0626	0.9500	0.8929	0.8848	0.8725
	SVR	1.0019	0.9568	0.9838	0.9442	0.9585	0.9551	0.9269
	ELM	0.9654	0.9360	0.9306	0.9289	0.9328	0.9351	0.9271
	SHL-DNN	**0.8342**	**0.8092**	**0.8041**	**0.8017**	**0.8043**	**0.8091**	**0.8057**
1-h	DNN	1.4944	1.3270	1.3093	1.2645	1.2252	1.2060	1.1988
	SVR	1.3409	1.3083	1.2829	1.2616	1.2559	1.2578	1.2439
	ELM	1.3095	1.2792	1.2761	1.2739	1.2643	1.2661	1.2644
	SHL-DNN	**1.1717**	**1.1625**	**1.1559**	**1.1496**	**1.1468**	**1.1447**	**1.1364**
2-h	DNN	2.0060	1.7768	1.7535	1.7181	1.7162	1.6947	1.6696
	SVR	1.8524	1.8180	1.7607	1.7544	1.7581	1.7411	1.7326
	ELM	1.8174	1.7830	1.7800	1.7775	1.7626	1.7605	1.7574
	SHL-DNN	**1.7059**	**1.6898**	**1.6832**	**1.6552**	**1.6565**	**1.6537**	**1.6406**

Table 3
MAPE of four models after different time scales on Ningxia data. The best performances are marked in boldface.

Time	Model	Training data of target domain						
		0.5-month	1-month	2-month	3-month	4-month	5-month	6-month
10-min	DNN	28.34	18.55	19.21	13.31	12.22	12.77	11.97
	SVR	20.23	18.65	19.08	19.75	18.57	18.65	18.98
	ELM	12.33	11.84	11.77	11.77	11.75	11.70	11.63
	SHL-DNN	**10.35**	**9.93**	**9.99**	**9.26**	**9.07**	**8.74**	**8.01**
30-min	DNN	33.69	23.70	24.23	19.02	18.33	18.20	18.13
	SVR	22.50	22.40	22.42	21.39	21.25	21.95	21.10
	ELM	18.61	18.02	17.96	17.94	17.84	17.80	17.76
	SHL-DNN	**17.07**	**16.55**	**16.04**	**16.03**	**16.22**	**15.99**	**15.99**
1-h	DNN	35.19	29.83	29.32	26.30	25.36	25.55	25.10
	SVR	30.70	29.82	28.78	27.97	28.39	28.32	27.50
	ELM	26.03	25.66	25.58	25.32	25.23	25.20	25.18
	SHL-DNN	**25.35**	**24.96**	**24.34**	**24.27**	**24.78**	**23.92**	**23.85**
2-h	DNN	47.49	41.31	40.08	36.83	36.43	35.86	35.74
	SVR	43.17	42.14	39.76	39.02	39.19	38.98	38.19
	ELM	**37.15**	**36.91**	**36.75**	36.51	**36.25**	36.09	36.17
	SHL-DNN	38.42	37.66	37.97	**35.66**	36.42	**35.46**	**34.70**

The input data should be normalized before they are employed in the model. After the learning process, the output of the model should be inverse-normalized before computing the prediction errors.

4.2. Learning a common model is useless

First, we conduct an experiment to discuss whether it is effective to simply combine the training data from different farms into

Table 4
MAE of four models after different time scales on Jilin data. The best performances are marked in boldface.

Time	Model	Training data of target domain						
		0.5-month	1-month	2-month	3-month	4-month	5-month	6-month
10-min	DNN	1.4439	0.8577	0.7745	0.5122	0.6320	0.5475	0.5171
	SVR	0.7388	0.7050	0.6078	0.5313	0.4659	0.4668	0.4757
	ELM	0.6056	0.5946	0.5876	0.5835	0.5834	0.5831	0.5818
	SHL-DNN	**0.4382**	**0.4306**	**0.4273**	**0.4241**	**0.4179**	**0.4157**	**0.4100**
30-min	DNN	1.5403	0.9678	0.9462	0.7597	0.8508	0.7752	0.7747
	SVR	0.8994	0.8596	0.7972	0.7563	0.7367	0.7383	0.7383
	ELM	0.8119	0.8157	0.8089	0.8046	0.8032	0.8036	0.8032
	SHL-DNN	**0.7200**	**0.7218**	**0.7198**	**0.7173**	**0.7146**	**0.7134**	**0.7098**
1-h	DNN	1.6433	1.1317	1.0987	0.9749	1.0377	0.9865	0.9890
	SVR	1.0667	1.0264	0.9759	0.9538	0.9573	0.9545	0.9486
	ELM	1.0149	1.0067	1.0012	0.9998	0.9997	0.9996	0.9965
	SHL-DNN	**0.9499**	**0.9463**	**0.9418**	**0.9387**	**0.9380**	**0.9377**	**0.9318**
2-h	DNN	1.7950	1.3994	1.3370	1.2746	1.3110	1.2833	1.3029
	SVR	1.3108	1.3243	1.2597	1.2353	1.2381	**1.2389**	1.2359
	ELM	1.3290	1.3116	1.2919	1.2914	1.2903	1.2933	1.2863
	SHL-DNN	**1.2673**	**1.2483**	**1.2363**	**1.2339**	**1.2347**	1.2441	**1.2319**

Table 5
MAPE of four models after different time scales on Jilin data. The best performances are marked in boldface.

Time	Model	Training data of target domain						
		0.5-month	1-month	2-month	3-month	4-month	5-month	6-month
10-min	DNN	32.47	30.53	24.08	16.74	16.18	16.36	13.86
	SVR	23.03	22.72	21.44	17.03	14.25	14.35	15.39
	ELM	15.98	15.81	15.66	15.35	15.43	15.31	15.30
	SHL-DNN	**12.74**	**12.85**	**12.43**	**12.83**	**12.59**	**12.33**	**11.80**
30-min	DNN	36.91	31.28	30.36	25.49	25.81	23.81	**21.17**
	SVR	28.61	26.63	26.53	24.53	23.21	23.55	23.71
	ELM	24.00	23.94	23.40	23.45	22.27	**22.31**	22.31
	SHL-DNN	**23.86**	**23.88**	**22.32**	**22.66**	**21.90**	22.71	22.70
1-h	DNN	40.25	36.51	37.21	33.52	32.49	31.77	30.26
	SVR	35.50	34.08	32.94	32.40	31.13	31.46	31.19
	ELM	35.09	33.99	32.96	31.80	**30.75**	**29.75**	**29.76**
	SHL-DNN	**34.78**	**33.88**	**32.16**	**31.76**	31.27	31.19	31.54
2-h	DNN	46.49	46.14	45.59	44.98	43.99	43.18	43.18
	SVR	42.34	42.17	42.60	41.84	40.84	40.62	40.20
	ELM	**41.54**	**40.92**	**40.92**	**39.87**	**39.53**	**39.51**	**38.56**
	SHL-DNN	42.02	41.88	41.38	40.47	41.71	40.85	40.42

Table 6
MAE of four models after different time scales on Inner Mongolia data. The best performances are marked in boldface.

Time	Model	Training data of target domain						
		0.5-month	1-month	2-month	3-month	4-month	5-month	6-month
10-min	DNN	1.5538	1.1241	0.8235	0.8346	0.8145	0.7968	0.7886
	SVR	1.0665	0.9536	0.9010	0.8609	0.8344	0.8175	0.8119
	ELM	0.9351	0.9280	0.9138	0.9100	0.9127	0.9123	0.9094
	SHL-DNN	**0.7057**	**0.6987**	**0.6926**	**0.6871**	**0.6801**	**0.6815**	**0.6798**
30-min	DNN	1.7875	1.3862	1.1651	1.1528	1.1750	1.1350	1.1292
	SVR	1.3006	1.2329	1.1932	1.1904	1.1909	1.1663	1.1620
	ELM	1.2118	1.2070	1.2021	1.1939	1.1916	1.1899	1.1920
	SHL-DNN	**1.0947**	**1.0979**	**1.0918**	**1.0845**	**1.0850**	**1.0808**	**1.0738**
1-h	DNN	1.8171	1.5809	1.4433	1.4370	1.4307	**1.4000**	1.4039
	SVR	1.5279	1.4901	1.4801	1.4856	1.4730	1.4586	1.4635
	ELM	1.4403	1.4302	1.4339	1.4278	1.4267	1.4242	1.4253
	SHL-DNN	**1.3805**	**1.3763**	**1.3750**	**1.3776**	**1.3748**	1.4076	**1.3752**
2-h	DNN	2.1031	1.8739	1.7586	**1.7448**	**1.7500**	**1.7429**	**1.7425**
	SVR	1.8418	1.8178	1.7803	1.7766	1.7693	1.7763	1.7625
	ELM	1.7734	1.7691	1.7633	1.7616	1.7608	1.7530	1.7556
	SHL-DNN	**1.7660**	**1.7638**	**1.7515**	1.7590	1.7534	1.7498	1.7510

one training set. In Model 1, the training set comprises 5-month data from each farm. After training, four 1-month data-sets are used as the testing sets. The result of each farm is compared with the separate model by using its own 5-month data. It is denoted by Model 2. They all conduct on a two-hidden-layer deep neural network model. Figs. 6 and 7 show the prediction errors of the two methods for wind farms in Ningxia and Jilin. It is clear that for MAPE, shown in theses two figures, the values of Model 2 are lower

Table 7
MAPE of four models after different time scales on Inner Mongolia data. The best performances are marked in boldface.

Time	Model	Training data of target domain						
		0.5-month	1-month	2-month	3-month	4-month	5-month	6-month
10-min	DNN	38.56	27.13	18.91	18.74	18.20	18.46	17.63
	SVR	25.97	23.13	20.77	19.82	19.19	18.83	18.65
	ELM	19.33	19.04	18.63	18.53	18.54	18.50	18.46
	SHL-DNN	**15.68**	**15.29**	**15.29**	**14.56**	**14.47**	**15.33**	**14.27**
30-min	DNN	43.17	33.63	27.24	26.14	26.24	25.53	25.81
	SVR	29.58	28.70	28.29	27.74	27.09	27.68	26.09
	ELM	25.65	25.50	25.35	25.20	25.09	25.07	25.05
	SHL-DNN	**25.94**	**25.62**	**24.17**	**23.96**	**23.96**	**23.36**	**22.26**
1-h	DNN	44.17	38.09	34.54	33.40	32.75	**32.49**	32.62
	SVR	35.27	34.97	34.29	34.29	34.15	34.95	33.43
	ELM	35.04	34.86	34.95	33.89	32.71	32.69	32.68
	SHL-DNN	**34.70**	**33.59**	**33.97**	**32.43**	**32.30**	32.59	**32.56**
2-h	DNN	50.90	46.46	44.52	42.06	43.08	42.46	42.40
	SVR	47.16	46.12	45.39	45.53	44.99	44.52	43.89
	ELM	**44.17**	**43.94**	43.84	**42.04**	**41.95**	**41.85**	**41.84**
	SHL-DNN	44.72	44.15	**43.80**	43.07	42.89	43.47	43.30

than those of Model 1. For some other indexes, the effects of the two methods are similar.

The results show that, although the amount of training data increases, the prediction accuracy is not higher and occasionally even negative. Therefore, learning a common model by simply using mixed data from all domains is ineffective. For this reason, we propose the shared-hidden-layer model shown in Fig. 5.

4.3. *Training strategies*

We design a series of experiments using the same architecture shown in Fig. 5 but with different training strategies. All strategies predict wind speeds for the next 8-h for the sake of comparison. The detailed training processes of the three strategies are described in Section 3.3.

For every farm, we use 5-month data for training and following 1-month data for testing. The results of two farms are shown in Figs. 8 and 9. From the figures, we can see that the second and third training methods are clearly better than the first. Because the first method does not involve labeled data in the shared hidden layers, it differs from the other two methods. This indicates that the label information is important for effective training. From the results, the supervised hidden-layer training is more significant for shorter-term wind forecasting. The prediction errors are close for a longer period of time. When comparing the last two strategies, the third fine-tunes the output layers by using separate data rather than all data, which is more reasonable. Although the experimental results are similar, the third method consistently outperforms the second. For MAPE, it is clear that the last strategy is the best. Therefore, we choose this Supervised-Finetune method for the training strategy in the following experiments.

4.4. *Transfer to small data-set farm*

Now, assume that a wind farm has a small volume of data. The three tests conducted are described in this section. They all predict the wind speeds in the target domain for the next 8-h as a comparison. Table 1 shows the different settings of three tests.

Figs. 10–12 show the experimental results for the different data-sets. We use four methods for each test. The first, SVR, indicates the results of support vector regression with linear kernel using only the target data. The second, DNN, represents the results of a two-hidden-layer deep neural network using the target data, which are fine-tuned after DSA pre-training. The third, ELM, is the results of extreme learning machine with the same kernel as SVR using the target data [39]. The forth, denoted as SHL-DNN, is our designed model, and uses the third training strategy described in the previous section.

From the figures, we can conclude that SVR is typically better than DNN, transfer learning is generally better than both SVR and DNN especially for shorter time scale prediction, and ELM performs better for MAPE in longer time while worse than SHL-DNN in other results. This indicates that SVR and ELM performs better than DNN when data is lack. When transferring data from other farms, SHL-DNN can discover common features of various wind farms from different locations. Therefore, transfer learning for the target domain which has insufficient amount of data is a very effective method.

4.5. *Size of target domain increases*

In this section, we examine the effect of model transferring when different amounts of target domain training data are available. Three tests on different data-sets are conducted similar to those described in the previous section. The difference here is the amount of target data. Tables 2–7 summarize the results of different measured error values for different domains. Only two evaluation criteria (MAE and MAPE) for the prediction errors are shown here owing to a lack of article space. Tables 2 and 3 are for Ningxia, Tables 4 and 5 are for Jilin and Tables 6 and 7 are for Inner Mongolia. The tables show that by using the transferred SHL-DNN, most of the results outperform the other three methods, which do

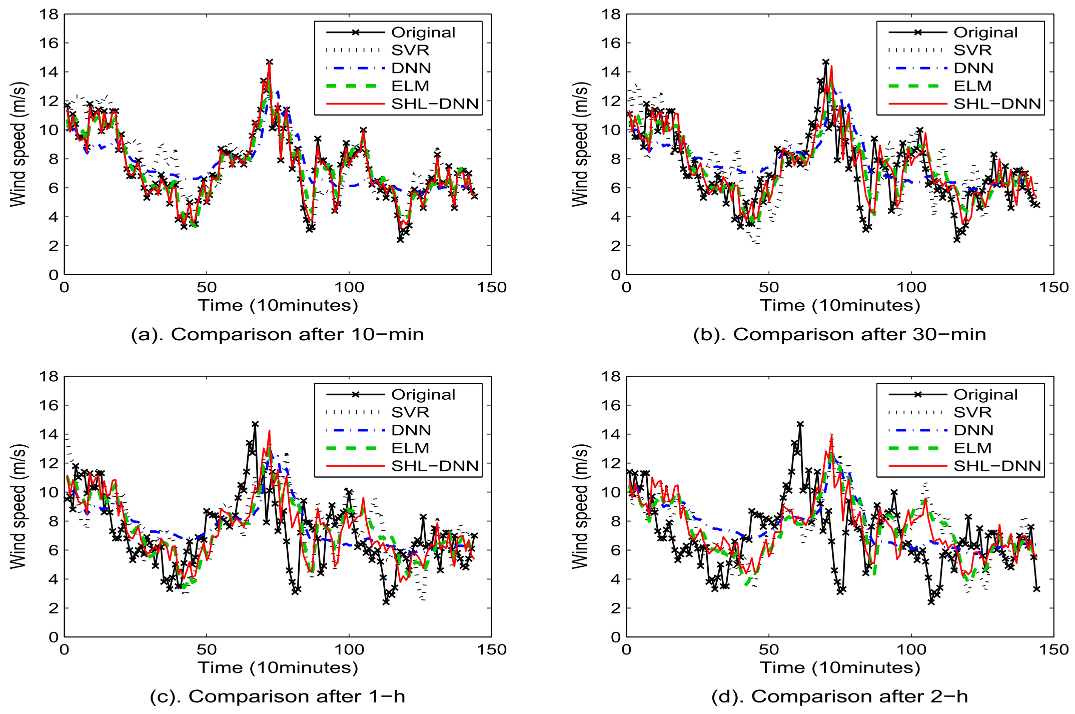

Fig. 13. Comparison between the original wind speeds and three models prediction results of half-month target training data on Ningxia data.

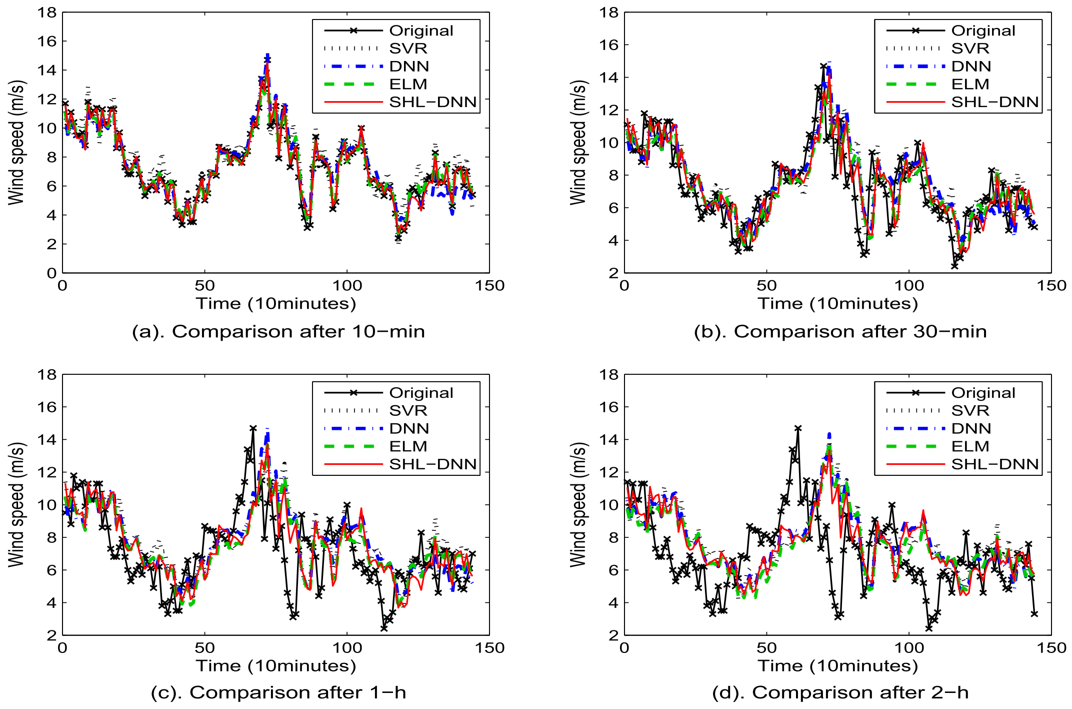

Fig. 14. Comparison between the original wind speeds and three models prediction results of 3-month target training data on Ningxia data.

not use such a transfer. Particularly for the next 10- and 30-min wind speed prediction, our proposed model is significant better than them when the amount of target data is small. It is also noted that as more information is available on the target domain, the importance of the transferred knowledge decreases, especially for forecasting at a longer time scale. For MAE, the most results of our method show the best performances. But for MAPE, the other models especially ELM are better for prediction results after 2-h. Hence, it can be concluded that if sufficient training data of the target domain are available, there is no need transferring data from other domains, which may occasionally decreases the prediction accuracy.

For test 1, Figs. 13–15 show a comparison of the original wind speeds and the prediction results of the four models on Ningxia data. The figures depict the wind speeds for different amounts of target training data. The target domains for the four models are made up of 0.5-, 3- and 6-month data, respectively. To better illustrate the consequences, only 1-day data are displayed. The sub-

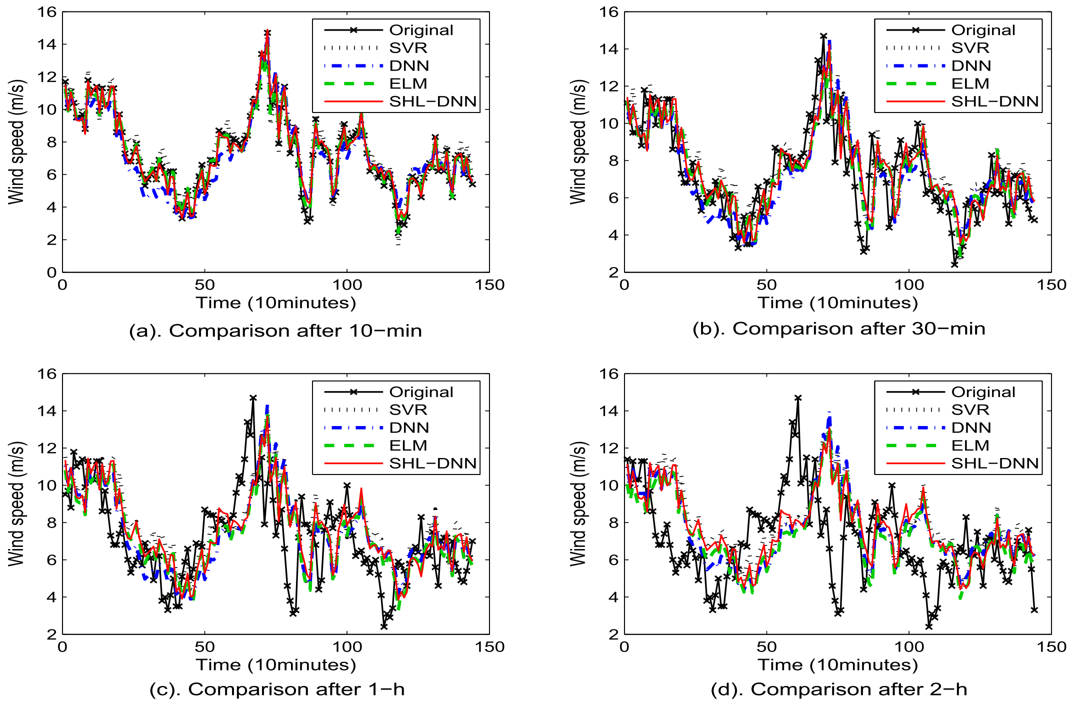

Fig. 15. Comparison between the original wind speeds and three models prediction results of 6-month target training data on Ningxia data.

graphs of each image depict the wind speed prediction after different time scales, i.e., 10-min, 30-min, 1-h and 2-h. The time series illustrated in Fig. 13 support the significant improvement in the output after the application of transfer learning to the wind speed predictions. The effects shown in Figs. 14 and 15 are not very clear. They further illustrate the conclusion that we have made in last paragraph. We can also see that in each figure, for longer time scales, the forecasting accuracy decreases.

5. Conclusions

In this paper, a shared-hidden-layer DNN architecture in which the hidden layers are shared across the domains and the output layers are different in each domain, is used for transfer learning. The model fuses the data from different wind farms in various regions. The results show that the proposed architecture is highly beneficial for a wind farm with insufficient training data. Our results also demonstrate that when there is not lots of training data, DNN models may be performed worse than other shallow models, such as SVR and ELM. When data is sufficient, it is effective when combining SDA unsupervised pre-training and supervised fine-tuning strategies. Furthermore, we also show if the training set of the target domain is increased, knowledge transfer eventually becomes less necessary.

For building the deep neural networks in this study, we use data from only four wind farms located in northern China. Even though the variation characteristics of wind at different sites are different because of weather, terrain, topography and so on, our transfer model is still proved to be effective. What remains to be investigated is whether this model can cover a wider range of regions, and whether data from many different wind farms located in different areas can build a shared model with separate parts for the learning of each individual farm. If it helps, this will be very meaningful, and we are therefore going to investigate the problem in our future work.

Acknowledgments

This work was supported by National Program on Key Basic Research Project (973 Program) under Grant 2012CB215201, National Natural Science Foundation of China (NSFC) under Grants 61432011, 61222210, 61170107, and New Century Excellent Talents in University under Grant NCET-12-0399.

References

[1] J.F. Manwell, J.G. McGowan, A.L. Rogers, Wind Energy Explained: Theory, Design and Application, John Wiley & Sons, 2010.

[2] X.Y. Yang, Y. Xiao, S.Y. Chen, Wind speed and generated power forecasting in wind farm, Chin. Soc. Electr. Eng. 25 (11) (2005) 1.

[3] F.O. Hocaoglu, M. Kurban, The effect of missing wind speed data on wind power estimation, in: Intelligent Data Engineering and Automated Learning, Springer, 2007, pp. 107–114.

[4] G. Giebel, L. Landberg, J. Badger, K. Sattler, H. Feddersen, T.S. Nielsen, H.A. Nielsen, H. Madsen, Using ensemble forecasting for wind power, in: Proc. Conf. European Wind Energy Association, Madrid, Spain, 2003.

[5] G. Giebel, L. Landberg, C. Bjerge, M. Donovan, A. Juhl, K. Gram Hansen, H. Waldl, T. Pahlke, J. Giebhardt, M. Rebbeck, et al., Cleverfarm-First results from an intelligent wind farm, in: Proc. Conf. European Wind Energy Association, Madrid, Spain, 2003.

[6] U. Focken, J. Jahn, M. Schaller, Transformer congestion forecast based on highly localized wind power, in: Talk on the 8th International Workshop on Large-scale Integration of Wind Power into Power Systems as Well as on Transmission Networks for Offshore Wind Farms, Bremen, 2009, pp. 14–15.

[7] M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, Z. Yan, A review on the forecasting of wind speed and generated power, Renew. Sustain. Energy Rev. 13 (4) (2009) 915–920.

[8] L. Landberg, Short-term prediction of the power production from wind farms, Wind Eng. Industrial Aerodynamics 80 (1) (1999) 207–220.

[9] G. Giebel, R. Brownsword, G. Kariniotakis, M. Denhard, C. Draxl, The state-of-the-art in short-term prediction of wind power: A literature overview, in: ANEMOS. Plus, 2011.

[10] W. Guoyang, X. Yang, W. Shasha, Discussion about short-term forecast of wind speed on wind farm, Jilin Electr. Power 181 (5) (2005) 4–21.

[11] M. Khalid, A. Savkin, A method for short-term wind power prediction with multiple observation points, IEEE Trans. Power Syst. 27 (2) (2012) 579–586.

[12] M. Bilgili, B. Sahin, A. Yasar, Application of artificial neural networks for the wind speed prediction of target station using reference stations data, Renew. Energy 32 (14) (2007) 2350–2360.

[13] T. Barbounis, J. Theocharis, Locally recurrent neural networks for long-term wind speed and power prediction, Neurocomputing 69 (4) (2006) 466–496.

[14] M. Carolin Mabel, E. Fernandez, Analysis of wind power generation and prediction using ANN: a case study, Renew. Energy 33 (5) (2008) 986–992.

[15] G. Li, J. Shi, On comparing three artificial neural networks for wind speed forecasting, Appl. Energy 87 (7) (2010) 2313–2320.

[16] S. Salcedo Sanz, A.M. Perez Bellido, E.G. Ortiz Garcia, A. Portilla Figueras, L. Prieto, F. Correoso, Accurate short-term wind speed prediction by exploiting diversity in input data using banks of artificial neural networks, Neurocomputing 72 (4) (2009) 1336–1341.

[17] E.G. Ortiz Garcia, S. Salcedo Sanz, A.M. Perez Bellido, J. Gascon Moreno, J.A. Portilla Figueras, L. Prieto, Short-term wind speed prediction in wind farms based on banks of support vector machines, Wind Energy 14 (2) (2011) 193–207.

[18] M. Mohandes, T. Halawani, S. Rehman, A.A. Hussain, Support vector machines for wind speed prediction, Renew. Energy 29 (6) (2004) 939–947.

[19] S. Salcedo Sanz, A.M. Perez Bellido, A. Portilla Figueras, L. Prieto, et al., Short term wind speed prediction based on evolutionary support vector regression algorithms, Expert Syst. Appl. 38 (4) (2011) 4052–4057.

[20] J. Zhou, J. Shi, G. Li, Fine tuning support vector machines for short-term wind speed forecasting, Energy Convers. Manag. 52 (4) (2011) 1990–1998.

[21] Z. Guo, J. Zhao, W. Zhang, J. Wang, A corrected hybrid approach for wind speed prediction in Hexi Corridor of China, Energy 36 (3) (2011) 1668–1679.

[22] E. Cadenas, W. Rivera, Wind speed forecasting in three different regions of mexico, using a hybrid ARI-MACANN model, Rnewable Energy 35 (12) (2010) 2732–2738.

[23] N. Chen, Z. Qian, I. Nabney, X. Meng, Wind power forecasts using Gaussian processes and numerical weather prediction, IEEE Trans. Power Syst. 29 (2) (2014) 656–665.

[24] A. Haque, M. Nehrir, P. Mandal, A hybrid intelligent model for deterministic and quantile regression approach for probabilistic wind power forecasting, IEEE Trans. Power Syst. 29 (4) (2014) 1663–1672.

[25] A. Togelou, G. Sideratos, N. Hatziargyriou, Wind power forecasting in the absence of historical data, IEEE Trans. Sustain. Energy 3 (3) (2012) 416–421.

[26] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in: Proc. 28th Int. Conf. Machine Learning, 2011, pp. 513–520.

[27] C. Plahl, R. Schluter, H. Ney, Cross-lingual portability of Chinese and English neural network features for French and German LVCSR, in: IEEE Workshop on Automatic Speech Recognition and Understanding, 2011, pp. 371–376.

[28] S. Thomas, S. Ganapathy, H. Hermansky, Multilingual MLP features for low-resource LVCSR systems, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 2012, pp. 4269–4272.

[29] J.T. Huang, J. Li, D. Yu, L. Deng, Y. Gong, Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 2013, pp. 7304–7308.

[30] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, J. Dean, Multilingual acoustic models using distributed deep neural networks, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 2013, pp. 8619–8623.

[31] N.T. Vu, D. Imseng, D. Povey, P. Motlicek, T. Schultz, H. Bourlard, Multilingual deep neural network based acoustic modeling for rapid language adaptation, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 2014, pp. 7639–7643.

[32] F.O. Hocaoglu, O.N. Gerek, M. Kurban, A novel wind speed modeling approach using atmospheric pressure observations and hidden Markov models, Wind Eng. Industrial Aerodynamics 98 (8) (2010) 472–481.

[33] D.C. Ciresan, U. Meier, J. Schmidhuber, Transfer learning for Latin and Chinese characters with deep neural networks, in: Proc. IEEE Int. Conf. Neural Networks, 2012, pp. 1–6.

[34] T. Amaral, L. Silva, L.A. Alexandre, J. Marques, J. Santos, Transfer learning using rotated image data to improve deep neural network performance, in: Image Analysis and Recognition, Springer, 2014, pp. 290–300.

[35] L. Deng, D. Yu, Deep Learning for Signal and Information Processing, Microsoft Research Monograph, 2013.

[36] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I.J. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde Farley, et al., Unsupervised and transfer learning challenge: a deep learning approach, in: Proc. ICML Unsupervised and Transfer Learning, 2012, pp. 97–110.

[37] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proc. Inter. Conf. Machine Learning, ACM, 2008, pp. 1096–1103.

[38] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009), 1–127.

[39] G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans, Syst. Man, Cybern. Part B: Cybern. 42 (2) (2012) 513–529.