Deep Neural Network Based Feature Representation for Weather Forecasting

James N.K. Liu[1], **Yanxing Hu**[1], **Jane Jia You**[1] , **and Pak Wai Chan** [2]
[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[2]Hong Kong Observatory, 134A Nathan Road, Kowloon, Hong kong

Abstract— *This paper concentrated on a new application of Deep Neural Network (DNN) approach. The DNN, also widely known as Deep Learning(DL), has been the most popular topic in research community recently. Through the DNN, the original data set can be represented in a new feature space with machine learning algorithms, and intelligence models may have the chance to obtain a better performance in the "learned" feature space. Scientists have achieved encouraging results by employing DNN in some research fields, including Computer Vision, Speech Recognition, Natural Linguistic Programming and Bioinformation Processing. However, as an approach mainly functioned for learning features, DNN is reasonably believed to be a more universal approach: it may have the potential in other data domains and provide better feature spaces for other type of problems. In this paper, we present some initial investigations on applying DNN to deal with the time series problem in meteorology field. In our research, we apply DNN to process the massive weather data involving millions of atmosphere records provided by The Hong Kong Observatory (HKO)*[1]*. The obtained features are employed to predict the weather change in the next 24 hours. The results show that the DNN is able to provide a better feature space for weather data sets, and DNN is also a potential tool for the feature fusion of time series problems.*

Keywords: Deep Neural Network, Stacked Auto-Encoder, Weather Forecasting, Feature Representation

1. Introduction

Deep Neural Networks (DNNs) or Deep Learning is the general term of a series of multi-layer architecture neural networks that are trained with the greedy layer-wise unsupervised pre-training algorithms[1], [2], [3]. Albeit controversial, DNNs have won great success in some fields including Computer Vision, Speech Recognition, Natural Linguistic Programming and Bioinformation processing. By applying the greedy layer-wise unsupervised pre-training mechanism, DNNs can reconstruct the original raw data set, in other words, DNNs can "learn" features with Neural Networks(NNs) instead of selecting features manually that we did traditionally[4]. And the intelligence models, like classifiers or regressors usually can obtain higher accuracy and better generalization with the learned features.

As its name suggested, DNN is a kind of NNs that structured by multiple layers. The word "Deep" indicates that such NN contains more layers than the "shallow" ones, which mainly includes the most widely used three-layer Feed Forward NNs in the past 30 years. Actually, multi-layer NN is not a new conception, some earlier studies have been conducted since 1990s[5], [6], but the successful implementation of multi-layer NNs was not realized until the provision of the novel Layer-wise unsupervised Pre-training mechanism by Hinton in 2006 that is employed to solve the training difficulties efficiently [1].

Although theoretically, a shallow NN with three layers trained with Back-Propagation(BP) has been proved that can approximate any nonlinear functions with arbitrary precision [7], when the number of hidden neurons is limited, the learning ability of shallow NNs may not be enough and poor generalization may be expected when using an insufficiently deep architecture for representing some functions. The significance of "deep" is that compared with shallow models, NN with deep architecture can provide a higher learning ability: functions that can be compactly represented by a deep architecture might be required to handle an exponential number of computational elements (parameters) to be represented by a deep architecture. More precisely, functions that can be compactly represented by a depth k architecture might require an exponential number of computational elements to be represented in a depth $k-1$ architecture [3].

Based on the Layer-wise unsupervised Pre-training mechanism, DNN can map the raw data set from the original feature space into a learned feature space layer by layer in the training process. In each layer, the unsupervised training may provide a kind of regularization to the data set and minimize the variance in each layer. Therefore, in the finally obtained feature spaces, classifiers or regressors have chances to obtain higher accuracy and better generalization. In some research areas, including computer vision [1], Speech recognition [8], Natural Linguistic Programming [9], and Bioinformation, DNNs have been reported achieving great success in the past five years.

The objective of our investigations is to explore the potential of DNN in other research domains. In previous research,

[1]http://www.hko.gov.hk/contente.htm

we note that for time series problem, a good representation of original feature space may be helpful for the applied model to get better performance [10]. Meanwhile, in time series problem, the correlations among features are obviously but not easy to be identified. If we can analyze the correlations and represent the features properly, the prediction accuracy is expected to be improved, and DNNs could be a reasonable and suitable tool to analyze the time series features. In this investigation, we apply a DNN model to predict the weather change in the next 24-hour period with a big data set. The massive data involving millions of weather records is provided by The Hong Kong Observatory (HKO). Our training method is to use the latest proposed greedy layer-wise unsupervised pre-training algorithm followed by a supervised fine-tuning. In detail, we choose a revised Auto-Encoder algorithm to build the network [11], the DNN is used to learn the features from the larger volume of raw data, and we evaluate the learned features according to the prediction accuracy. The contribution and significance of our investigation demonstrate that: compared with the classical models, NNs with Deep architectures can improve the prediction accuracy in weather forecasting domain, and our initial study gives the results that can show the potential of DNN on time series problems.

2. Weather Prediction Problem

The changes of climate that could impact people's daily life, therefore people never cease their efforts on predicting the trend of weather changes. Unlike data sets in other domain, weather data has some particularities. Specifically, there is season-to-season, and year-to-year variability in the trend of weather data. The cycle could be multi-month, multi-season or multi-year, and the main difficulty of investigations is to capture all the possible cycles.

Many significant research efforts are utilized to develop weather forecasting methods including artificial intelligence technologies that have been accepted as appropriate means for weather forecasting and reported encouraging results since 1980s [12], [13], [14]. Among many different intelligence models, single variable time series regression is the most fundamental and most widely applied one in weather forecasting, especially short-term predictions. Since our initial investigation is an exploration on the application of DNN in the area of weather forecasting, in this paper, we mainly concentrate on employing DNN to represent the feature space for single variable time series regression problem.

Generally speaking, for a certain variable, the objective of single variable time series regression is to find the relationship between its status in a certain future time point and its status in a series of past time points, and estimate its future status via:

$$v_t = f(v_{t-1}, v_{t-2}, \ldots, v_{t-n}) \quad (1)$$

The function f, can be obtained by employing different intelligence models such as Linear Regression, Generalized Linear Model, Autoregressive Integrated Moving Average Mode, etc.

In our investigation, we focus on forecasting four kinds of weather information, including temperature, dew points, Mean Sea Level Pressures(MSLP) and wind speed, in the next few hours. We will input the raw data sets into our DNN model, the input n-dimensional vector is composed of the status in $(t-1)th$, $(t-2)th$, $\ldots$, $(t-n)th$ time points, then we use the DNN to represent these status, and employ a regressor to estimate the status in tth time point. We hope the seasonal cycles can be captured via massive volume of data by the superior learning ability of the multi-layer structured NN.

3. Greedy Layer-wise Unsupervised Pre-training and Aotuencoder DNN

The essential challenge in training deep architectures is to deal with the strong dependencies that exist during training between the parameters across layers [15]. Multi-layer NNs have more parameters than shallow NNs. Moreover, in a multi-layer NN, due to the non-convexity of the complex model, the optimization with traditional BP training approach may fall in a local minimum rather than global minimum. This may bring poor generalization to the model.

This problem isn't well solved until Hinton et al. introduced Deep Belief Network (DBN) that greedily trained up one layer with a Restricted Boltzmann Machine (RBM) at a time in 2006 [1]. Shortly after, strategies for building deep architectures from related variants were proposed by Bengio [16] and Ranzato[17]. They solved the training problem of deep NN in two phases: in the first phase, unsupervised pre-training, all layers are initialized using this layer-wise unsupervised learning signal; in the second phase, fine-tuning, a global training criterion (a prediction error, using labels in the case of a supervised task) is minimized. Such training approach is called the Greedy Layer-wise Unsupervised Pre-training. Fig.1 [15] shows the comparison among different training methods for NNs with deep architectures. There is a family of training models categorized into the family of Greedy Layer-wise Unsupervised Pre-training approaches, In our investigation, with the consideration of the attribute type of the weather data, i.e., the collected data are all real numbers, we choose the Stacked Auto-Encoder to build the deep architecture of our NN model.

The Stacked Auto-Encoder, as its name suggested, is a stacked architecture NN that applies Auto-Encoder in each layer. In NN, a single "neuron" is a computational unit that taken as input vector $X = x_1, x_2, \ldots, x_n$ (and a $+1$ intercept term), and outputs $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$ with a nonlinear function $f : \Re \mapsto \Re$. W is the weight matrix that stands for the connection among different

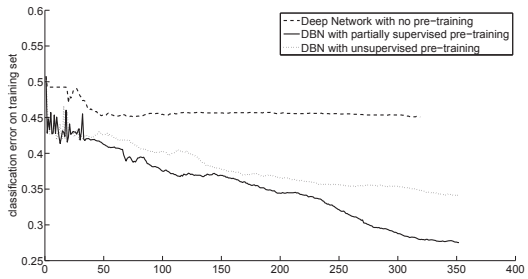

Fig. 1: Training classification error vs training iteration on DNNs, which shows the optimization difficulty for DNNs and the advantage of pre-training methods.

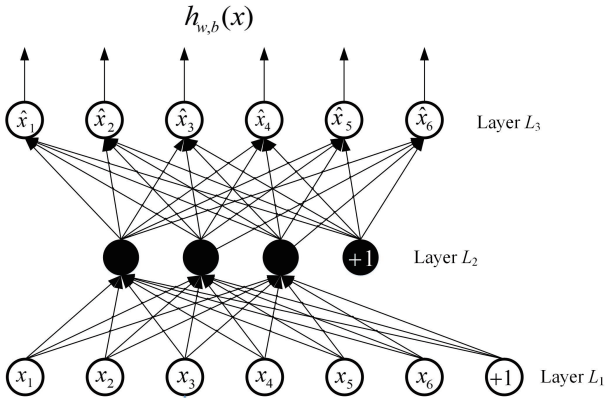

Fig. 2: An illustration of Auto-Encoder Algorithms. Layer L_1 is the input layer, and L_3 is the output layer. Via hidden layer L_2, we hope to represent the information x in layer L_1, so that the output $\hat{x}$ in L_3 can approximate the raw data x.

neurons in the network. In most of cases, sigmoid function $f(z) = \frac{1}{1+\exp(-z)}$ is employed. A typical Auto-Encoder tries to learn a function $h_{w,b}(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x . The identity function seems a typically trivial function trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data [17], e.g., for a data set, suppose that the original samples are collected from a 100-dimensional feature space, i.e. $x \in \Re^{100}$, set that there are 50 hidden units in the hidden layer, based on the requirement $h_{w,b}(x) \approx x$, the network is forced to learn a compressed representation of the input. That is, given only the vector of hidden unit activations $a^{(2)} \in \Re^{50}$, it must try to reconstruct the 100-dimensional input x. An illustration of Auto-Encoder is shown in Fig.2. If the inputs were completely random, each x_i comes from an I.I.D. Gaussian independent of the other features, then this compression task would be very difficult. But if there is a certain structure hidden in the data, for example, if some of the input features are correlated, such as in the feature space of time series analysis, then this algorithm will be able to discover some of those correlations.

The loss function of Auto-Encoder is:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - x^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \quad (2)$$

where m is the number of training samples. The objective of the Auto-Encoder is to minimize Eq.(2) in order to make sure that the output $h_{W,b}(x^{(i)})$ can approximate the raw data $x^{(i)}$ as far as possible. The second term in Eq.(2) is a regularization term (also called a weight decay term) controled by the weight decay parameter λ that tends to decrease the magnitude of the weights, and helps prevent overfitting. We can minimize Eq.(2) by *gradient descent* to compute the configuration of the network.

Consequently, we need to combine Auto-Encoders layer by layer with a stacked structure to build the DNN. For each layer, we use an Auto-encoder to train the parameters in this layer, and then have these layers combined together. Specifically, in the training process of each layer, as shown in Fig.1, the input vectors have to pass through three layers, and the vectors in hidden layers (layer L_2, and for simplicity, we call the vectors in layer L_2 as the transformed vectors of the initially input vectors) are representations of the input vectors and can be used to reconstruct the input vectors. Thus, in every layer of the deep NN, the input of the current layer is the output of the previous layer, then we train the input data via an Auto-Encoder, and use the transformed vectors as the output of the current layer. Fig.4 shows the detailed mechanism of stacked Auto-Encoder based DNN. We can see that through a DNN, the raw data can be represented in new feature spaces layer by layer. In other words, DNN can learn features from the original data sets. Consequently, we apply any proper intelligence models with the learned features.

4. Experimental Results and Analysis

In our experiments, we concentrate on the evaluation of the learned features: four types of weather data sets, are employed and simulated, and the results are compared between models using raw features and models using represented features.

4.1 Weather Data Collection and Pre-processing

The HKO has provided great support to our investigation. Based on our collaboration with HKO, a massive volume of high quality real weather data could be applied in our experiment. Historical weather data sets, including the temperature, dew points, MSLP and wind speed data are employed in

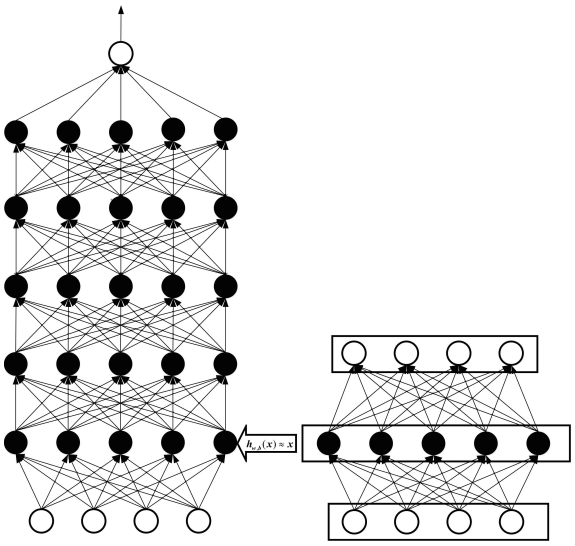

Fig. 3: A 5-hidden-layer DNN with Stacked Auto-Encoder method, by which each layer is greedily pre-trained with an unsupervised Auto-Encoder to learn a nonlinear transformation of its input (the output of the previous layer) that captures the main variations in its input, i.e. $h_{W,b}(x) \approx x$.

our model. The time range of the data sets is almost 30-year long, which covers the period from January 1, 1983 to December 31, 2012. In detail, the number of records in each data set is more than 260,000 respectively.

Unlike the temperature (measured with degree Celsius), dew points (measured in degree Celsius)and MSLP (measured in hectopascal(hPa)) data which have only one dimension , the wind speed data has two dimensions: the polar coordinate for the wind direction (measured with degree angle) and the speed (measured with meters per second). Moreover, for a certain time points, the direction of the air motion is not stable, i.e. the wind direction at that time point is not fixed. Such condition is denoted as "variable" in the raw data. Therefore, according to the requirement of our algorithm, we have to do some pre-processing on the data sets.

Different from the temperature (as Fig.4), dew points (as Fig.5) and MSLP (as Fig.6) data which are one-dimensional scalar quantities, the wind speed data (in a fixed horizontal plane) is a vector quantity that has two dimensions in the polar coordinate (as Fig.7), i.e. the angle to show its direction and the speed to measure the magnitude in this direction[18]. However, since our model is focused on single variable time series problems, we have to transform the data set to satisfy the model's requirement. According to the physical significance of the two dimensions, we denote the angle as θ and the speed as v to obtain:

$$v^0 = cos\theta \cdot v \tag{3}$$

where v^0 is the vector components of the wind speed in

Fig. 4: The distribution of temperature data in the last week of the data set

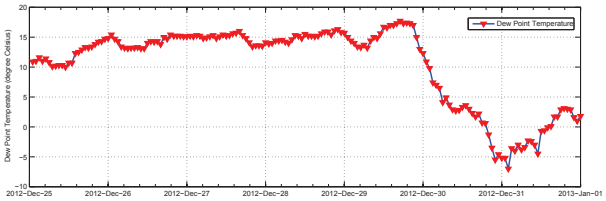

Fig. 5: The distribution of dew points temperature data in the last week of the data set

0 degree angle direction (as Fig.8). Thus, what we actually simulate is the time series of the speed components of the air motion in 0 degree angle direction. Moreover, there are about 3% wind speed data with the direction valued as "variable", for such condition, we consider it as a missing value in the data set and use the average value of the wind direction in its previous time point and its next time point to replace the value "variable".

4.2 Experiment on Temperature/Dew Points Temperature Forecasting

In our first experiment, we use a 4-layer DNN model to predict the temperature and dew points temperature in the next time point. The temperature data set records the real-time temperature, and the dew points temperature is a more complex quantity: the dew point is the temperature at which the water vapor in air at constant barometric pressure condenses into liquid water at the same rate at which it evaporates [19]. At temperatures below the dew point, water will leave the air. From Fig.4 and Fig.5, we can see that both of the two data sets show a smooth changing trend and are not as unstable as other two data sets. We may also observe

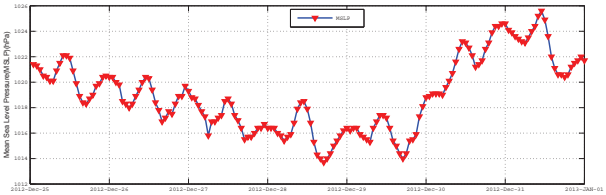

Fig. 6: The distribution of MSLP data in the last week of the data set

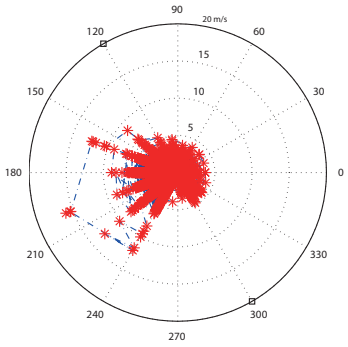

Fig. 7: The distribution of wind speed data in polar coordinate

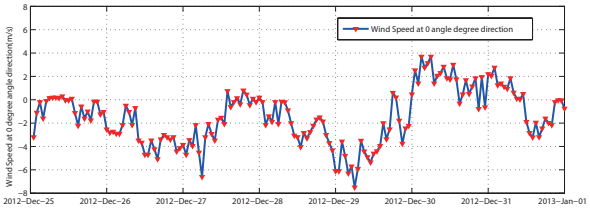

Fig. 8: The distribution of wind speed at a fixed diection

that the temperature data set and the dew points data set show very similar curves, which indicated that there may be a high relevance between these two data sets. However, the dew points curve is a little more fluctuating than the temperature cure, which imply that other factors may have effect on its values.

The objective of our experiments is to try to use the 7-day hourly records to forecast these two temperature related data in the next 24-hour period respectively, The whole network is with an input layer, two stack-organized Auto-Encoder layers, and the top layer which uses Support Vector Regression (SVR) to output the prediction results [20], [21]. Table I shows the parameter configuration of our experiment.

In the experiments, the training set has more than 230,000 records, and about 26,000 samples are selected as testing set. the ratio between training set and testing set is 9:1. The result is compared with Classical SVR. Note that the parameters in the Classical SVR are set as same as in the top SVR

Table 1: The parameter configuration of the experiments

Parameter	Value
Number of neurons in hidden layer 1	82
Number of neurons in hidden layer 2	40
λ	0.05
Learning rate	0.01
Max Iteration	400
Parameters in SVR	Default as LibSVM [22]

Table 2: The comparison of temperature prediction by SVR and DNN

Model	NMSE	DS	R^2
Classical SVR	2.179e-2	0.75	0.872
DNN with SVR in Top Layer	8.117e-3	0.82	0.915

Table 3: The comparison of dew points prediction by SVR and DNN

Model	NMSE	DS	R^2
Classical SVR	2.132e-2	0.75	0.856
DNN with SVR in Top Layer	9.552e-3	0.80	0.901

layer of the DNN model. In our experiment, we apply three criteria including Normalised Mean Square Error (NMSE), Directional symmetry (DS) and R^2 to evaluate the prediction results. Table II and Fig.9 give the result of temperature forecasting, and the results of dew points forecasting is given as Table III and Fig.10.

From Fig.9 and Fig.10 we can observe that both SVR and DNN can simulate the two temperature related real data sets very well after training with a massive volume of data. For the temperature data, the predicted results almost coincide with the real data, the similar predict results are also obtained in the dew points data. We can see that after training with a massive volume of data set, the SVR can efficiently simulate the temperature related data sets. Especially, the DNN isn't troubled with the overfitting problem when it is trained with the large data set.

Results shown in Table II and Table III demonstrate the positive role of the DNN in the simulation tasks. As we

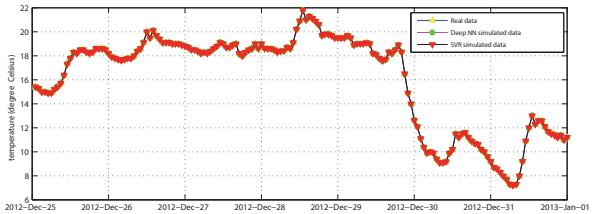

Fig. 9: The results of temperature prediction for the date in the last week

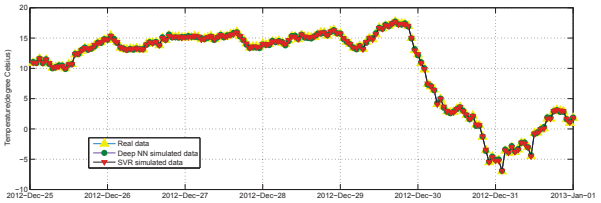

Fig. 10: The results of dew points temperature prediction for the date in the last week

Table 4: The comparison of MSLP prediction by SVR and DNN

Model	NMSE	DS	R^2
Classical SVR	6.26	0.70	0.851
DNN with SVR in Top Layer	2.13	0.72	0.924

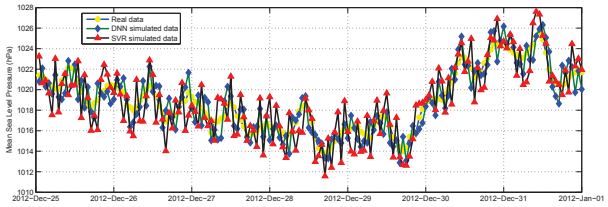

Fig. 11: The results of temperature prediction for the date in the last week

Table 5: The comparison of wind speed prediction by SVR and DNN

Model	NMSE	DS	R^2
Classical SVR	0.3721	0.72	0.831
DNN With SVR in Top Layer	0.2522	0.83	0.891

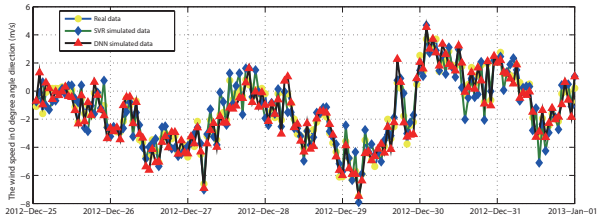

Fig. 12: The results of temperature prediction for the date in the last week

discussed in above, both of the two models have the same configuration in SVR part, the only difference is that in the DNN, we represent the raw features, and use the new features to train the SVR in the top layer. We can see that the DNN model greatly reduces the NMSE(the NMSE in SVR model has already been very small, but after the feature representation/granulation, the NMSE becomes even smaller). Also, the higher R^2 values are obtained. These results demonstrate that, with the represented features via a deep NN, the SVR model in the top layer can learn the raw data much better.

4.3 Experiment on MSLP Forecasting

In real applications, temperature simulation (in short term) is not a challenging task since that the change of temperature in short term is relatively stable. Therefore, although the DNN can work very well on simulation of temperature related data sets as shown in the previous experiments, the academic and practical significance is weak, experiments on more unstable data set is necessary to confirm the result.

In the following experiment, we apply the MSLP data to verify our model. The MSLP is the atmospheric pressure at sea level or (when measured at a given elevation on land) the station pressure reduced to sea level assuming that the temperature falls at a lapse rate of 6.5 K/km in the fictive layer of air between the station and sea level [23]. Observing Fig.6, we may find that the curve of MSLP data is more fluctuating than that of the temperature data. So it is reasonable to apply MSLP data to verify our model further. We adopt the parameter configuration of the previous experiments. Table V and Fig.11 give the results.

From the results shown in Table IV and Fig.11, we shall observe that there are obvious error values between the real data and the predicted data, which is different from the condition in previous experiments that the predicted curves almost coincide with the actual curves. However, the high R^2 value shows that both of the SVR and DNN model can explain more than 85% of the errors in the simulation, and the DNN model can avoid overfitting when training with the large data set.

Again, compared with the SVR model, the DNN model shows its advantages in this experiment. The NMSE is reduced and the R^2 value is obviously improved, such experimental results demonstrate that with the learned feature, the SVR model can provide higher accuracy in MSLP prediction.

4.4 Experiment on Wind Speed Forecasting

As shown in Fig.4, Fig.5, Fig.6 and Fig.8, the change of wind speed data is much more stochastic than other three data sets. Accordingly, the simulation of wind speed data is more difficult and has more academic and practical significance. To improve the learning ability of our model, we have made some modifications on the configuration of the DNN, a hidden layer with more neurons is added in the DNN, and the sparsity setting is given in this added layer. The results are shown in Table V and Fig.12.

Fig.12 shows the simulation results of the wind speed data in the last week of the data set. From Fig.12, we can observe that after training with a big volume of wind speed data, the model can capture the main trend of changes, and the DNN can give a better performance than simply using SVR. Inspecting the criteria in Table V, DNN can return a lower NMSE and higher R^2 value.

There are two special points which should be noticed in this experiment. (1) From the distribution of the MSLP data set and the wind speed data set, we know that it is more difficult to predict the wind speed than to predict MSLP; however, in our simulation, the latter experiments can obtain lower NMSE value. This fact implies that the added hidden layer is able to improve the learning ability of the whole network.(2) Note that a much higher DS value is obtained in DNN, this maybe caused by the fact that

features generated via DNN may have the largest possible variation, and such fact shows that the principle of DNN may be considered as an advanced form of Principal Component Analysis (PCA)[24].

Our experiments only make comparison between classical SVR and Stacked Auto-Encoder DNN with SVR in the top layer. Actually, some other models can also be applied to deal with weather data related time series problem. However, the main objective of our investigation is to attest the models' performance with the new represented features. The results demonstrate that compared with the raw features, the obtained features can explain the principle of the raw data set better. Moreover, the DNN can be combined with many other models, and the learned features can be employed to improve the performances of most models in computational intelligence field.

5. Conclusion, Limitation and Future Work

In our investigation, we explore an approach that using a novel computational intelligence technology to process massive volume of weather data. The proposed DNN model may represent the features of the raw weather data layer by layer , and experimental results show that the obtained features can improve the performances of classical computational intelligence models. The contribution of our investigation is significant: we give an approach that using computational intelligence method to learn features for weather forecasting, and our experiments demonstrate that the DNN algorithm also has the potential on time series problem.

Our initial investigation is to explore the use of DNN to deal with the large volume of weather data. Therefore, limitations also exist. The relation between the number of hidden layers and the output accuracy should be quantified, also, the number of hidden neurons should be optimized.

The main future work of our investigation is that, we will try to employ our model on more difficult weather data, such as rain fall data set; and moreover, we will continue exploring the theoretical principle of computational intelligence, especially, we will try to give the mathematical explanation of the DNN.

Acknowledgment

The authors would like to acknowledge the partial support of the CRG grants G-YL14 and G-YM07 of The Hong Kong Polytechnic University.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[3] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc., 2009, vol. 2, no. 1.

[4] X. Wang and Q. He, "Enhancing generalization capability of svm classifiers with feature weight adjustment," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2004, pp. 1037–1043.

[5] J. Schmidhuber, "Curious model-building control systems," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on*. IEEE, 1991, pp. 1458–1463.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[8] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012.

[9] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning." *Journal of Machine Learning Research-Proceedings Track*, vol. 27, pp. 17–36, 2012.

[10] J. N. Liu and Y. Hu, "Application of feature-weighted support vector regression using grey correlation degree to stock price forecasting," *Neural Computing and Applications*, pp. 1–10, 2013.

[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 9999, pp. 3371–3408, 2010.

[12] S.-M. Chen and J.-R. Hwang, "Temperature prediction using fuzzy time series," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 30, no. 2, pp. 263–275, 2000.

[13] K. Kwong, M. H. Wong, J. N. Liu, and P. Chan, "An artificial neural network with chaotic oscillator for wind shear alerting," *Journal of Atmospheric and Oceanic Technology*, vol. 29, no. 10, pp. 1518–1531, 2012.

[14] W.-C. Hong, "Rainfall forecasting by technological machine learning models," *Applied Mathematics and Computation*, vol. 200, no. 1, pp. 41–57, 2008.

[15] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[16] S. M. Miller, Y. Geng, R. Z. Zheng, and A. Dewald, "Presentation of complex medical information: Interaction between concept maps and spatial ability on deep learning," *International Journal of Cyber Behavior, Psychology and Learning (IJCBPL)*, vol. 2, no. 1, pp. 42–53, 2012.

[17] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. Lecun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[18] R. A. Pielke, *Mesoscale meteorological modeling*. Academic Press, 2002.

[19] J. Ancsin, "Dew points, boiling points and triple points of," *Metrologia*, vol. 9, no. 1, p. 26, 1973.

[20] V. Vapnik, *The nature of statistical learning theory*. Springer, 2000.

[21] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[22] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[23] T. Moses, G. Kiladis, H. Diaz, and R. Barry, "Characteristics and frequency of reversals in mean sea level pressure in the north atlantic sector and their relationship to long-term temperature trends," *Journal of Climatology*, vol. 7, no. 1, pp. 13–30, 1987.

[24] K. Yang and C. Shahabi, "A pca-based similarity measure for multivariate time series," in *Proceedings of the 2nd ACM International Workshop on Multimedia Databases*. ACM, 2004, pp. 65–74.