Deep Learning for Wind Speed Forecasting in Northeastern Region of Brazil

Anderson Tenório Sergio, and Teresa B. Ludermir
Centro de Informática
Universidade Federal de Pernambuco, UFPE
Recife, Brasil
{ats3,tbl}@cin.ufpe.br

***Abstract*—Deep Learning is one of the latest approaches in the field of artificial neural networks. Since they were first proposed in mid-2006, Deep Learning models have obtained state-of-art results in some problems with classification and pattern recognition. However, such models have been little used in time series forecasting. This work aims to investigate the use of some of these architectures in this kind of problem, specifically in predicting the hourly average speed of winds in the Northeastern region of Brazil. The results showed that Deep Learning offers a good alternative for performing this task, overcoming some results of previous works.**

Keywords—deep learning; neural networks; time series forecasting; wind forecasting

I. INTRODUCTION

One of the main goals of the research in Artificial Intelligence is to emulate the efficiency and robustness with which the human brain interprets sensory input and processes it in a useful way for daily activities. During the more than 50 years of studies in this field, various designs and architectures have been proposed, tested and used successfully in a practical way in several branches of knowledge. An example of these models is artificial neural networks, having the feed-forward network as their best-known architecture. However, all the proposed models undergo the so-called "curse of dimensionality" [1].

A possible solution to this problem is data pre-processing by reducing its dimensionality, usually performed by humans [2]. However, this solution can be challenging and highly dependent on the task [3]. Moreover, there is no evidence that the human brain processes similarly the vast amount of data to which it is exposed. This organ takes advantage of an intricate hierarchy of modules and, over time, learns to represent observations based on regularities that they exhibit. These findings were obtained through the work of Lee et al. [4] [5]. They studied how the visual cortex of the mammalian brain operates. The neocortex, for example, does not require pre-processed sensory signals.

Given to this scenario, it was developed a new subarea in artificial intelligence that dealt with computer models for information representation that exhibited characteristics that were similar to the ones of the neocortex. This subarea is often called Deep Learning (DL). The Deep learning algorithms can be considered as learning processes that find multiple levels of representation, and more abstract features of the data are represented by higher levels [6]. The more abstract representations can be more useful in extracting information for classifiers or predictors [7]. In addition to this, the learned intermediate characteristics can be shared among different tasks [8].

Over decades, many researchers have unsuccessfully tried to train neural networks with multiple deep layers [9] [10]. Networks were often trapped in local minima when initialized with random weights. As depth increased, a good generalization became even more difficult. Experiments have shown that the results of deep neural networks with randomly initialized starting weights obtain worse results than neural network with one or two hidden layers [11] [12]. In 2006, Hinton et al. found that the results of a deep neural network could be significantly improved when pre-trained with a non-supervised learning algorithm, one layer after another, starting from the first layer [13]. This work booted the area now known as Deep Learning. At the beginning of the research in this subarea, in general, algorithms had the following characteristics: unsupervised learning of representations to pre-train each of the layers; unsupervised training, one layer at the time, being the representation learned at each level the input to the next layer; using supervised training for fine tuning of all pre-trained layers, as well as one or more additional layers dedicated to the production of predictions.

The models that make up Deep Learning can be applied in various problems, both in academy and in industry, i.e. in speech recognition and signal processing [14], object recognition [15], natural language processing [16] and transfer learning [17]. Even with some works available (reviewed in section III), the use of Deep Learning in time series forecasting has received less attention. A time series can be defined as a chronological sequence of observed data of any task or periodic activity in such fields as engineering, biology, economics or social sciences [18].

This study aims to investigate some of the Deep Learning models in wind speed forecasting in the northeastern region of Brazil. The prediction of these values can significantly reduce the operating difficulties in a wind power plant, as well as in traditional sources of energy, since it is possible to integrate efficient models of wind power generation forecasting with other electrical generation systems [19].

II. Deep Learning

A. Deep Belief Networks

Deep Belief Networks (DBN) are neural networks that follow a generative probabilistic model, first introduced in [13]. Generative models provide a probabilistic distribution on data and labels, providing the estimation of P(Observation| Label) and P(Label|Observation). Discriminative models, like most conventional neural networks, only provide P(Label|Observation). DBN aim to mitigate some of the training problems of conventional neural networks: the need of much labeled data in the training set; slow training and inadequate learning techniques for selection parameters that lead to local minima.

DBNs are based on Sigmoid Belief Networks, a generative multilayer neural network proposed before the appearance of Deep Learning, trained with variational approaches [20]. An example of a Sigmoid Belief Network can be seen in Figure 1. Network is represented as a directed graphical model and each random variable node is represented by directed arcs, indicating a direct dependence. The observed data is x and the hidden factors in the k level are the elements of vector h^k. The parameterization of conditional distributions (in the downward direction) is similar to activation of conventional neuron networks, given by Equation 1. h_i^k is the binary activation of the hidden node i in layer k, h^k is the vector $(h_1^k, h_2^k, \dots)$, where $x = h^0$.

$$P\left(h_i^k = 1 \middle| h^{k+1}\right) = sigm(b_i^k + \sum_j W_{i,j}^{k+1} + h_j^{k+1}) \quad (1)$$

The bottom layer generates a vector x in the input space. Considering multiple levels, the generative model of a Sigmoid Belief Network is given by Equation 2.

$$P(x, h^1, \dots, h^l) = P(h^l)\left(\prod_{k=1}^{l-1} P(h^k|h^{k+1})\right)P(x|h^1) \quad (2)$$

In this case, $P(x)$ is intractable in practice, except in very small models. The upper layer is then factored given by $P(h^l) = \prod_i P(h_i^l)$.

Deep Belief Networks are quite similar to Sigmoid Belief Networks. In DBN, probabilities are given by components called Restricted Boltzmann Machines (RBM). Deep Belief Network architecture can be seen in Figure 2.

RBM, represented in Figure 3, are non-directed graphical models without connections between nodes in the same layer. They are composed of input units (or visible) x_j and hidden units h_i, allowing the calculation of $P(h|x)$ and $P(x|h)$. An RBM is normally trained by a learning algorithm that trains each layer at a time in a greedy way, building more abstract representations of the original data through $P(h^k|x)$. Hinton called this algorithm Contrastive Divergence [21].

In comparison with other one layer models, RBM are more attractive as building blocks due to their easy training. According to the Contrastive Divergence algorithm, a vector v is presented to the visible units during the training phase. These values are propagated to the hidden units. On the other hand, the entries are stochastically calculated aiming at the reconstruction of the original data, through a process known as Gibbs Sampling. The correction of weights is given by the difference in the correlation of the hidden and visible entries activations. The training time is reduced considerably and each added layer increases the probability of obtaining the training data.

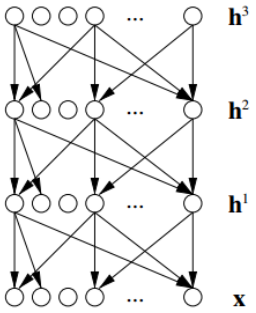

Figure 1: Sigmoid Belief Network.

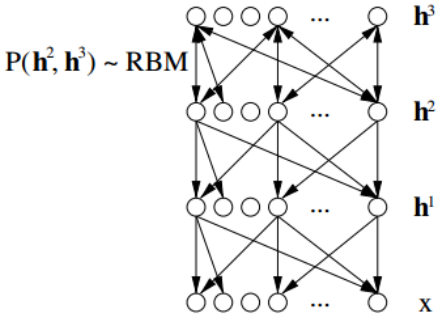

Figure 2: Deep Belief Network.

After this pre-training, DBN may admit fine tuning for better discriminative performance. The adjustment is accomplished by the use of labeled data and another training algorithm, such as the well-known backpropagation. In this case, a set of labels is attached to the upper layers of the network.

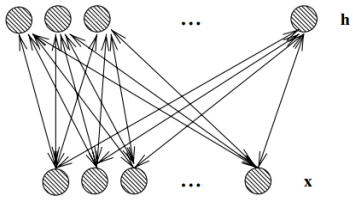

Figure 3: Restricted Boltzmann Machine.

B. Stacked Autoencoders

Autoencoders are a type of neural network also known as Diabolo networks [22]. Overall, training a self-encoder is easier than training an RBM. Self-encoders have been used as building blocks in Deep Learning architectures, in which each level is trained separately [11].

An autoencoder is trained for encoding an input x for some representation *c(x)* so that it can be reconstructed from this representation. The decoding function *f(c(x))* gives the reconstruction of the network, usually a vector of numbers obtained through a sigmoid. The expectation is for *c(x)* to be a distributed data representation that captures the main factors of variations.

According to [8], the procedure to train Stacked Autoencoders is similar to the process performed in Deep Belief Networks: (i) Train the first layer as an autoencoder in order to minimize the reconstruction error of the original input, in an unsupervised way. (ii) The outputs of the hidden units are used as inputs for another layer, also trained as an autoencoder. These two steps do not require labeled data. (iii) Iterate step two to boot the desired number of additional layers. (iv) Use the output of the last hidden layer as input for a supervised layer and randomly initialize its parameters or use it in a supervised manner. (v) Perform fine-tuning of all parameters of this deep architecture through a supervised criterion. Alternatively, fold all autoencoders on a very deep autoencoder and perform fine-tuning in the overall reconstruction error.

The idea is that autoencoders must have low reconstruction error in the training examples, but high reconstruction error in most other configurations. Autoencoders can be regularized to avoid simply learning the identity function. An example is the so-called Denoising Auto-Encoders [23] that uses versions of the input data with noise. The model used in this work is known as Stacked Denoising Auto-Encoders (SDAE).

III. Deep Learning for Time Series Forecasting

Deep Learning models were originally used in classification and pattern recognition problems. After that, such models started to be applied in various machine learning tasks, among which, time series forecasting. Although literature has some works on DL with series (some of them reviewed below), there are still no conclusive results about the role of pre-training on this model and the relationship between architecture and the dimensionality of the data. The dimensionality is usually provided by the lags of the time series.

Romeo et al. [24] used deep neural networks to predict temperatures series. The authors used Stacked Denoising Autoencoders as pre-training. The experiment took into account the fine tuning that occurred only in the last layer or in all network layers. The performance was improved when compared to systems without pre-training, but not as much as in other types of problems. This could be justified by the characteristic of the series used and the low-dimensional data.

Kuremoto et al. [25] used an only one hidden layer Deep Belief Network to predict a competition time series. According to the authors' experiments, the performance of DBN was better than the one of models such as MLP (Multilayer Perceptron), Bayesian learning and ARIMA. In DBN, performance was improved with the use of differential data. The hyperparameters model (lag of the time series, the number of neurons in the hidden layer and learning rates) was optimized by PSO (Particle Swarm Optimization). Despite having only one layer, the model is considered deep learning by having pre-training.

Chen et al. [26] also used a Deep Belief Net to forecast a series containing drought index data in an Asian river basin. As DBN component, the authors used an RBM for continuous data (CRBM). Unlike the previous model, network was built with two hidden layers. The results were better than in a common MLP.

Chao et al. [27] used a DBN to predict a time series of exchange rate. In this case, CRBM were also used as building blocks of the model. The results were better than in the feed forward architectures.

IV. Case Study

Several parameters can directly influence the outcome of the forecast. Therefore, a strategy for the selection of these parameters is required. At this work, the following parameters were varied by a ramdom grid search: the number of neurons in the input layer of the network, the number of neurons in the first hidden layer, the number of neurons in the second hidden layer and whether there is or not any type of pre-training.

A time series may be formalized as a sequence of scalar random observations $s = s_0, \dots, s_{i-1}, s_i, s_{i+1}$. The lag of the series is given by the delay used to form the network training and testing data. Then, time series forecasting means predict a future value of the sequence, given by $\overline{s}_{i+1} = F[s_i, s_{i-k}, s_{i-2k}, \dots, s_{i-(d-1)k}]$. d is the lag, k is the step lag and F, in this work, is the neural network.

The number of neurons in the network input layer is related to the time series lag. The number of hidden layers and the number of neurons in each of them are associated with the complexity of the model, since it increases the number of free parameters. The fewer free parameters, the easier it is to train the model. The more complex the network, the greater is the risk of overfitting. This work has chosen to use two hidden layers in the network, since algorithms and architectures of Deep Learning are used.

In order to investigate the use of Deep Learning approaches in time series forecasting series, two types of pre-training were used in the experiment. In the first case, each of the layers was considered to be an autoencoder and noise was added to the input data, characterizing the model known as Stacked Denoising Autoencoders (SADE). In another approach, each of the layers was considered a Restricted Boltzmann Machine, characterizing the model known as Deep Belief Network (DBN). Depending on how the parameters were selected, some networks were trained without any pre-training, and could, thus, be considered a common MLP. The aim was to verify whether it would be worth carrying out the pre-training or not.

Three time series of the SONDA project [28] were used. The SONDA project keeps free databases of solar and wind energy resources in Brazil. They are:

- Belo Jardim (BJD): City in the state of Pernambuco. Series with 13176 patterns consisted of hourly average speed of wind obtained from the wind power

plant of Belo Jardim, from July 1, 2004 to December 31, 2005.

- São João do Cariri (SCR): City in the state of Paraíba. Series with 17520 patterns consisted of hourly average speed of wind obtained from the wind power plant of São João do Cariri, from January 1, 2006 to December 31, 2007.
- Triunfo (TRI): City in the state of Pernambuco. Series with 21936 patterns consisted of hourly average speed of wind obtained from the wind power plant of Triunfo, from July 1, 2004 to December 31, 2006.

Figure 4 shows the methodology used in this case study.

The values for *n*, *h1* and *h2* were defined empirically after exhaustive testing. The performance used for the selection of the best networks in the training phase is the mean square error (MSE), given by equation 3.

$$MSE = \frac{1}{N * P}\sum_{i=1}^{P}\sum_{j=1}^{N}(T_{ij} - L_{ij})^2 \quad (3)$$

In equation 3, *P* is the number of patterns in the data set, N is the number of output units of the network and T_{ij} and L_{ij} are respectively the desired output value and the value calculated by the i-th neuron of the output layer. In this experiment, *N* is given by 1, because it is a problem of time series prediction.

1. Select database. db = [BJD, SCR, TRI]
2. Define the possible values for the number of input neurons. n = [25, 50, 75, 100, 125, 150]
3. Define the possible values for the number of neurons in the first hidden layer. h1 = [25, 50, 75, 100]
4. Define the possible values for the number of neurons in the second hidden layer. h2 = [25, 50, 75, 100]
5. Define the type of pre-training. pt = [MLP (without pre-training), DBN, SDAE].
6. Using training database (75% of full database), execute 100 training runs randomly selecting one of the possible values for each parameter (n, h1, h2, pt). The pre-training, if any, is performed on all layers, except the output layer. Fine tuning is performed with Levenberg-Marquadt algorithm.
7. Select the 20 models with better performance in training phase.
8. Using the test database (25% of full database), check the performance of selected models.

Figure 4: Methodology of Case Study

To evaluate the network in the testing phase and in order to measure performance, besides using the MSE, the mean absolute error (MAE) and the percentage of absolute error (MAPE) were also used. MAE and MAP are given respectively by equations 4 and 5.

$$MAE = \frac{1}{N * P}\sum_{i=1}^{P}\sum_{j=1}^{N}|T_{ij} - L_{ij}| \quad (4)$$

$$MAPE = 100 * \frac{1}{N * P}\sum_{i=1}^{P}\sum_{j=1}^{N}\left|\frac{T_{ij} - L_{ij}}{T_{ij}}\right| \quad (5)$$

V. Results and Discussion

Next, the results will be shown and discussed. Table I presents the configuration and performance of the top 20 systems created for the series of Belo Jardim (BJD). Table II shows the mean and standard deviations in the performance of each approach at a test database. The best results are highlighted.

One can see that the best system was generated by an architecture without any kind of pre-training. On average, the performance of the MLP model was better. However, this comparison between the means must be made carefully, because the number of systems created with each of the possible architectures varies widely. An achievement of the Deep Learning approaches was the fact of having generated more systems among the top 20, 17 DBN models and 1 SDAE models.

TABLE I. 20 Best Systems Created for BJD

Model	N	h1	h2	MSE	MAE	MAPE
DBN	50	50	75	0.666960	0.626273	0.120186
DBN	25	25	25	0.681332	0.633652	0.120052
SDAE	25	25	25	0.711954	0.653798	0.122903
MLP	25	25	50	0.638798	0.617384	0.115075
DBN	25	25	50	0.681478	0.639206	0.122665
DBN	50	75	50	0.648109	0.620767	0.119127
DBN	25	75	75	0.678515	0.635774	0.120895
DBN	25	25	75	0.670442	0.630098	0.119028
DBN	50	25	75	0.678035	0.635109	0.120006
DBN	100	25	25	0.661876	0.628315	0.120674
DBN	25	50	25	0.672718	0.629412	0.118400
DBN	75	25	50	0.675832	0.635165	0.120369
DBN	50	50	50	0.664829	0.627536	0.119125
DBN	25	50	50	0.670523	0.627389	0.117496
DBN	50	75	25	0.644938	0.619416	0.117406
DBN	25	75	50	0.677723	0.636206	0.120341
DBN	25	50	75	0.711025	0.648855	0.123957
DBN	50	25	50	0.678427	0.631983	0.121384
MLP	25	75	25	0.599255	0.596238	0.112018
DBN	25	75	25	0.681167	0.639159	0.121880

TABLE II. Mean and Standard Deviation for BJD

Model	MSE	MAE	MAPE	Quantity
MLP	0.619027 (0.027961)	0.606811 (0.014953)	0.113546 (0.002162)	2
DBN	0.673172 (0.014611)	0.632019 (0.007177)	0.120176 (0.001719)	17
SDAE	0.711954 (0.000000)	0.653798 (0.000000)	0.122903 (0.000000)	1

Following the same idea, tables III and IV show the results for the series of São João do Cariri (SCR). Then the tables V and VI do the same for the series of Triunfo (TRI).

One can see that, in SCR database, systems with pre-training have obtained the best performance taking into consideration all the error measures. Regarding the average performance, DBN was better with MAE and MLP with MAPE and MSE. However, just one MLP was in the best systems set, against 19 DBN. It is worth to note that no SDAE was in the best group.

TABLE III. 20 BEST SYSTEMS CREATED FOR SCR

Model	n	h1	h2	MSE	MAE	MAPE
DBN	50	50	75	0.685671	0.640265	0.126682
DBN	25	25	25	0.662088	0.626428	0.123490
DBN	50	75	75	0.692896	0.646241	0.134538
DBN	75	75	25	0.694653	0.642838	0.137266
DBN	25	25	50	0.662638	0.630890	0.128272
DBN	50	25	75	0.688955	0.643765	0.126561
DBN	25	75	75	0.680738	0.637218	0.126756
DBN	25	25	75	0.662190	0.630120	0.122203
DBN	75	50	50	0.684201	0.640217	0.134845
DBN	50	25	50	0.690305	0.643674	0.133328
DBN	25	50	25	0.649658	0.621233	0.127053
DBN	50	25	25	0.671109	0.632003	0.136889
MLP	25	50	50	0.676899	0.638682	0.126585
DBN	25	50	50	0.667266	0.631705	0.126921
DBN	75	25	25	0.687672	0.637887	0.133360
DBN	75	50	25	0.684736	0.635869	0.127056
DBN	25	50	75	0.680012	0.639830	0.123427
DBN	25	75	50	0.666790	0.629922	0.125876
DBN	50	75	25	0.675534	0.635686	0.129938
DBN	100	75	75	0.687922	0.643723	0.138402

TABLE IV. MEAN AND STANDARD DEVIATION FOR SCR

Model	MSE	MAE	MAPE	Quantity
MLP	0.676899 (0.000000)	0.638682 (0.000000)	0.126585 (0.000000)	1
DBN	0.677633 (0.012804)	0.636290 (0.006776)	0.129624 (0.005062)	19
SDAE	-	-	-	0

For the Triunfo time series, the best performance according to the MSE, MAE and MAPE was achieved by the DBN models. As in the other two databases, most of the best systems were DBN, without any SDAE. The top 20 set contained 4 MLP. With respect to the average performance, better approach came from the DBN models according to all error measures.

Although in the BJD series the best performance was achieved by an architecture without any kind of pre-training, one can see that among the systems created for all databases, the DBN model has outperformed the others approaches in the vast majority of cases according to all error measures. So we can infer that, to predict the wind speed in the databases studied, the DBN model would be the best choice. Also, it is possible to conclude that in these same conditions, the model with worst performing would be SDAE. This Deep Learning architecture just figured once among the best systems created, in BJD series.

All tested systems had two hidden layers. Some time ago, this would be an impediment for numerical simulations. But with the growth of computing power, to train neural networks with more than one hidden layer is not as expensive as it once was. If caution is taken to avoid overfitting, the gain in adding another hidden layer can directly impact the performance of the systems. This can be proved when the results of this study are compared with other forecasts performed for these same weather series. Table VII compares the best performance of the models tested in this article with two similar approaches using Reservoir Computing optimization with evolutionary algorithms, one with Genetic Algorithm (GA) and another one with PSO (Particle Swarm Optimization).

TABLE V. 20 BEST SYSTEMS CREATED FOR TRI

Model	n	h1	h2	MSE	MAE	MAPE
MLP	25	25	25	1.941906	1.021253	0.110764
DBN	25	25	25	1.551106	0.951877	0.106038
DBN	50	75	25	1.496886	0.921589	0.100605
MLP	25	25	50	1.828327	1.023901	0.115217
DBN	25	25	50	1.575730	0.954733	0.105676
DBN	50	25	50	1.482294	0.926188	0.103646
DBN	100	75	50	1.521698	0.923593	0.102849
DBN	25	25	75	1.463582	0.909818	0.100964
DBN	50	50	50	1.635742	0.969299	0.104607
DBN	75	50	50	1.516546	0.930249	0.103804
DBN	50	75	75	1.509517	0.928221	0.102806
DBN	25	75	75	1.764116	1.001344	0.108077
MLP	25	50	50	1.754493	1.004082	0.110033
DBN	25	50	50	1.463823	0.911560	0.100764
MLP	25	75	50	1.735531	0.997910	0.107941
DBN	50	75	50	1.421144	0.898138	0.098984
DBN	25	50	75	1.563875	0.952422	0.105285
DBN	100	75	25	1.601268	0.966846	0.112077
DBN	50	25	25	1.488428	0.922586	0.101310
DBN	75	25	25	1.408690	0.896288	0.101205

TABLE VI. MEAN AND STANDARD DEVIATION FOR TRI

Model	MSE	MAE	MAPE	Quantity
MLP	1.815064 (0.093558)	1.011787 (0.012758)	0.110989 (0.003062)	4
DBN	1.529028 (0.087889)	0.935297 (0.028542)	0.103669 (0.003301)	16
SDAE	-	-	-	0

Even taking into account that the experiments have not been replicated and that a more sophisticated approach would be required to compare the results, one can see that the use of Deep Learning to predict the hourly average speed of the winds in the Northeast of Brazil achieved at least relevant results, even beating previous work in some performing measures. Even the models without pre-training (and therefore not classified as Deep Belief Nets or Stacked Denoising Autoencoders) may be considered as part of DL architectures, since they were trained with more than one hidden layer.

VI. CONCLUSIONS AND FUTURE WORKS

This work sought to investigate the use of Deep Learning to predict the hourly average wind speed in northeastern Brazil. The forecast of this information may economically impact the region because it can provide a better scaling of resources in wind power plants as well as more conventional energy

sources. Through the experiments and the analysis of the results, it was found that the methodology used produced models of neural networks with satisfactory performance in all studied databases. In some of them the proposed experiment showed the best results found in literature.

TABLE VII. COMPARISON WITH OTHER WORKS

Model	BJD		SCR		TRI	
	MAE	MAPE	MAE	MAPE	MAE	MAPE
This work	0.5962 (MLP)	0.1120 (MLP)	0.6212 (DBN)	0.1222 (DBN)	0.8962 (DBN)	0.1012 (DBN)
RC-GA-I [29]	0.62	0.1208	0.63	0.1329	0.87	0.0986
RC-GA-II [29]	1.08	0.2027	1.23	0.2475	1.71	0.2063
RC-GA-III [29]	0.66	0.1247	0.70	0.1370	0.90	0.1023
RC-PSO [30]	-	0.1783	-	0.1736	-	0.0853

Regarding the analyzed models, all of them had two hidden layers, which can then be classified as Deep Learning. The pre-training of weights, thus creating models like Deep Belief Networks and Stacked Denoising Autoencoders, presented different performances. DBN have obtained the best performance and SDAE the worst. However, all models, when compared with other works in literature, had relevant results.

A future work already underway is the combination of different generated predictions. An example close to this idea can be seen in [29]. Is this work, the authors combine several outputs with a Suppport Vector Machine. Another point to be investigated is the use of pre-training that would be more focused on dynamic data such as Conditional RBM. Other Deep Learning architectures can be tested.

Finally, the model parameters can be optimized by PSO or GA. As can be seen in [29] and [30], good results can be achieved when neural networks are optimized by this kind of algorithm.

REFERENCES

[1] R. Bellman, "Dynamic programming and Lagrange multipliers." Proceedings of the National Academy of Sciences of the United States of America 42.10 (1956): 767.

[2] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning-a new frontier in artificial intelligence research [research frontier]." Computational Intelligence Magazine, IEEE 5.4 (2010): 13-18.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern classification. John Wiley & Sons, 2012.

[4] T. S. Lee, and D. Mumford, "Hierarchical Bayesian inference in the visual cortex." JOSA A 20.7 (2003): 1434-1448.

[5] T. S. Lee, et al, "The role of the primary visual cortex in higher level vision." Vision research 38.15 (1998): 2429-2454.

[6] Y. Bengio, "Deep learning of representations: Looking forward." Statistical language and speech processing. Springer Berlin Heidelberg, 2013. 1-37.

[7] Y. Bengio, A. Courville, and P. Vincent. "Representation learning: A review and new perspectives." Pattern Analysis and Machine Intelligence, IEEE Transactions on 35.8 (2013): 1798-1828.

[8] Y. Bengio, "Learning deep architectures for AI." Foundations and trends in Machine Learning 2.1 (2009): 1-127.

[9] Y. Bengio, Y. LeCun, "Scaling learning algorithms towards AI." Large-scale kernel machines 34.5 (2007).

[10] P. E. Utgoff, D. J. Stracuzzi, "Many-layered learning." Neural Computation 14.10 (2002): 2497-2529.

[11] Y. Bengio, et al, "Greedy layer-wise training of deep networks." Advances in neural information processing systems 19 (2007): 153.

[12] H. Larochelle, et al, "Exploring strategies for training deep neural networks." The Journal of Machine Learning Research 10 (2009): 1-40.

[13] G. Hinton, S. Osindero, and Y-W. Teh, "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[14] F. Seide, L. Gang, and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." Interspeech. 2011.

[15] D. C. Ciresan, et al, "Deep, big, simple neural nets for handwritten digit recognition." Neural computation 22.12 (2010): 3207-3220.

[16] R. Collobert, et al, "Natural language processing (almost) from scratch." The Journal of Machine Learning Research 12 (2011): 2493-2537.

[17] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning." Unsupervised and Transfer Learning Challenges in Machine Learning, Volume 7 (2012): 19.

[18] R. Fildes, et al, "Forecasting and operational research: a review." Journal of the Operational Research Society 59.9 (2008): 1150-1172.

[19] R. de Aquino, et al, "Recurrent neural networks solving a real large scale mid-term scheduling for power plants." IJCNN. 2010.

[20] P. Dayan, et al, "The helmholtz machine." Neural computation 7.5 (1995): 889-904.

[21] G. Hinton, "Training products of experts by minimizing contrastive divergence." Neural computation 14.8 (2002): 1771-1800.

[22] H. Bourlard, and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition." Biological cybernetics 59.4-5 (1988): 291-294.

[23] P. Vincent, et al, "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[24] P. Romeu, et al, "Time-Series Forecasting of Indoor Temperature Using Pre-trained Deep Neural Networks." Artificial Neural Networks and Machine Learning–ICANN 2013. Springer Berlin Heidelberg, 2013. 451-458.

[25] T. Kuremoto, et al, "Time series forecasting using a deep belief network with restricted Boltzmann machines." Neurocomputing 137 (2014): 47-56.

[26] J. Chen, J. Qiongji, and J. Chao, "Design of Deep Belief Networks for Short-Term Prediction of Drought Index Using Data in the Huaihe River Basin." Mathematical Problems in Engineering 2012 (2012).

[27] J. Chao, F. Shen, and J. Zhao, "Forecasting exchange rate with deep belief networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[28] SONDA, Sistema de Organização Nacional de Dados Ambientais. http://sonda.cptec.inpe.br/, May 2005.

[29] A. A. Ferreira, T. B. Ludermir, and R. De Aquino, "An approach to reservoir computing design and training." Expert systems with applications 40.10 (2013): 4172-4182.

[30] A. T. Sergio, Otimização de Reservoir Computing com PSO [Reservoir Computing Optimization with PSO]. 2013. Dissertação (Mestrado em Ciências da Computação [Master's Thesis]) - Universidade Federal de Pernambuco.

[31] X. Qiu, et al, "Ensemble deep learning for regression and time series forecasting." IEEE SSCI 2014-2014 IEEE Symposium Series on Computational Intelligence-CIEL 2014: 2014 IEEE Symposium on Computational Intelligence in Ensemble Learning, Proceedings. Institute of Electrical and Electronics Engineers Inc., 2015.