Time series prediction and forecasting using Deep learning Architectures

ScuDo

Scuola di Dottorato ~ Doctoral School

WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Electronics and Telecommunication (29^{th}cycle)

Time series prediction and forecasting using Deep Learning Architectures

By

Sanam Narejo

******

Supervisor(s):

Prof. Eros Pasero

Doctoral Examination Committee:

Prof. Francesco Carlo Morabito, Referee, Universita Mediterranea Reggio Calabria, Italy.

Prof. Salvatore Vitabile, Referee, University of Palermo, Italy.

Prof. Giansalvo Cirrincione, Université de Picardie Jules Verne, France.

Prof. Alippi Cesare, Politecnico di Milano, Italy.

Prof. Maurizio Martina, Politecnico di Torino, Italy.

Politecnico di Torino

2018

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Sanam Narejo
2018

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving grand parents, my parents, my supportive husband and my daughter Noor.

Acknowledgements

At first and foremost, I would like to mention my Almighty Allah who provided me strength to achieve my goals for which I dreamed once. Afterwards, My gratitude goes to my Supervisor Prof. Eros Pasero for his support throughout the entire PhD journey. I would like to acknowledge him, for his thought-provoking assistance, technical knowledge of Neural Networks allowed me to progress alot and I thank him for all he taught me about this subject. I am privileged and I feel lucky to be one of his research student. Additionally, I would also like to thank my colleagues and my lab mates specially Fahad bin Muslim and Khurshid Aliev for their help. The sincerity of efforts and guidance that they have provided is ineffable.

I would also like to thank the HPC at Politecnico di Torino for providing the computational resources.

I am thankful to the higher education commission (HEC) Pakistan for funding my doctorate at the Politecnico di Torino. This is a great initiative by the government of Pakistan to create a pool of high quality researchers who can ultimately contribute to the prosperity of the nation.

I am thankful to two external reviewers Prof. Francesco Carlo Morabito and Prof. Salvatore Vitabile, who assisted in the review process and provided constructive comments on my work to make this thesis a significant contribution to the literature. I would also like to pay my gratitude to Prof. Giansalvo Cirrincione for his interest in my work.

I would also like to thank those who contributed directly or indirectly to my thesis including my family members.

Abstract

Nature brings time series data everyday and everywhere, for example weather data, physiological signals and biomedical signals, financial and business recordings. Predicting the future observations of a collected sequence of historical observations is called time series forecasting. Forecasts are essential, considering the fact that they guide decisions in many areas of scientific, industrial and economic activity such as in meteorology, telecommunication, finance, sales and stock exchange rates. Massive amount of research has already been carried out by researchers over many years for the development of models to improve the time series forecasting accuracy. The major aim of time series modelling is to scrupulously examine the past observation of time series and to develop an appropriate model which elucidate the inherent behaviour and pattern existing in time series. The behaviour and pattern related to various time series may possess different conventions and infact requires specific countermeasures for modelling. Consequently, retaining the neural networks to predict a set of time series of mysterious domain remains particularly challenging. Time series forecasting remains an arduous problem despite the fact that there is substantial improvement in machine learning approaches. This usually happens due to some factors like, different time series may have different flattering behaviour. In real world time series data, the discriminative patterns residing in the time series are often distorted by random noise and effected by high frequency perturbations. The major aim of this thesis is to contribute to the study and expansion of time series prediction and multistep ahead forecasting method based on deep learning algorithms. Time series forecasting using deep learning models is still in infancy as compared to other research areas for time series forecasting.Variety of time series data has been considered in this research. We explored several deep learning architectures on the sequential data, such as Deep Belief Networks (DBNs), Stacked AutoEncoders (SAEs), Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). Moreover, we also proposed two different new methods based on muli-

step ahead forecasting for time series data. The comparison with state of the art methods is also exhibited. For time series data the visualization of learnt filters reports meaningfull representations in terms of long term features (Trends) and short term(subtle changes). The research work conducted in this thesis makes theoretical, methodological and empirical contributions to time series prediction and multi-step ahead forecasting by using Deep Learning Architectures.

Contents

List of Figures **xi**

List of Tables **xiv**

1 Introduction **1**

1.1 Time Series Forecasting 3

1.2 Towards Machine learning, Neural Networks and Deep learning . . 5

1.3 Motivation and Aim 7

1.4 State of the art 8

1.5 Main Contributions 13

1.6 Publications 13

1.7 Organization of thesis 14

2 Deep Learning Architectures (Theoretical Concepts) **17**

2.1 Introduction 17

2.2 Deep Belief Network (DBN) 18

2.2.1 Restricted Boltzmann Machine (RBM) 20

2.2.2 RBM for Real Data 22

2.3 Stacked AutoEncoder (SAE) 23

2.4 Bells and whistles 26

3 Time Series Prediction using DBN **27**

3.1 Hyper-parameter Settings 27

3.1.1 Learning rate 28

3.1.2 Mini batch size 28

3.1.3 Number of iterations 28

3.1.4 Momentum 28

3.1.5 Regularization 29

3.1.6 Hidden layer dimensions 29

3.1.7 Computational resources 30

3.2 Internet Traffic prediction 30

3.2.1 Related Work 32

3.2.2 Dataset Description 32

3.2.3 Experimental Setup 33

3.3 Performance Validation 35

3.3.1 Lorenz behaviour 36

3.3.2 Mackey-Glass behaviour 37

3.3.3 Validation 37

3.3.4 Results, Analysis and Discussion 38

3.4 Meteorological Nowcasting 43

3.4.1 METEO Weather Station and NEMEFO 45

3.4.2 Air Temperature prediction 45

3.4.3 Relative Humidity prediction 49

3.4.4 Pressure prediction 52

3.4.5 Discussion 57

3.5 Conclusion 61

4 Hybrid Approach for multi step ahead forecasting **63**

4.1 Outline . . . 63
4.2 Research Method for Hybrid Approach . . . 64
4.3 Experimental Setup . . . 65
4.3.1 Feature Learning with DBN . . . 66
4.3.2 Nonlinear Autoregressive models for multistep forecasting . 71
4.4 Results and Analysis . . . 73
4.5 Conclusion . . . 78

5 EEG signal classification for eye state Identification using SAE 80
5.1 Introduction . . . 80
5.2 Problem Statement . . . 81
5.3 State of the Art . . . 82
5.4 Materials and Method . . . 83
5.4.1 Data Description . . . 84
5.4.2 Preprocessing and Feature Extraction . . . 86
5.4.3 Classification Models . . . 86
5.5 Results and Discussion . . . 90
5.6 Conclusion . . . 94

6 Multi step Rainfall forecasting using Temporal DBN 95
6.1 Rainfall Dataset . . . 97
6.2 Filtering and Noise Removal . . . 97
6.3 Feature Extraction . . . 98
6.4 Methodology . . . 99
6.5 Results and Discussion . . . 100
6.5.1 One step ahead forecasting . . . 100
6.5.2 Four steps ahead forecasting . . . 101

6.5.3 Eight steps ahead forecasting 101

7 Conclusions and Future Developments **104**

References **107**

List of Figures

2.1 A stack of Restricted Boltzmann Machines (RBMs), greedily trained as Deep Belief Network (DBN). 20

2.2 Graphical structure of RBM with visible layer and hidden layer . . . 21

2.3 Illustration of an AutoEncoder. 24

2.4 Iterative training of stacked Autoencoder. 25

3.1 Infastructure of an ISP 31

3.2 Deep Beleif Network for Model I 34

3.3 Deep Beleif Network for Model II 35

3.4 Deep Beleif Network for Model III 35

3.5 Time series of synthesized Lorenz signal 36

3.6 Time series of synthesized Mackey-Glass signal 37

3.7 Internet traffic time series 39

3.8 Unsupervised pretraining of hidden layers 39

3.9 Nonlinear representations as learnt features of hidden layers from Model I 41

3.10 Histogram of errors on trained data 42

3.11 Sixty samples of prediction estimated by Model I (from test data set) 42

3.12 METEO Weather station. 43

3.13 Illustration of Contributed Activity for Weather Nowcasting 44

3.14 Recorded months converted into Sine waveform 46
3.15 Recorded hours converted into Sine waveform 47
3.16 Actual and filtered temperature Series 48
3.17 DBN-RBM for Temperature prediction 49
3.18 Actual and filtered humidity series 51
3.19 DBN-RBM for Humidity prediction 52
3.20 Actual and filtered pressure time series 53
3.21 DBN-RBM for Pressure prediction 54
3.22 Histograms: showing weights distribution after pretraining step. . . 55
3.23 Histograms: showing weights distribution after fine tuning step. . . 56
3.24 Close view of target and predicted temperature with eighty samples from test dataset. 58
3.25 Close view of target and predicted humidity with eighty samples from test dataset. 58
3.26 Close view of target and predicted pressure with fifty samples from test dataset. 59

4.1 Proposed Method for Multistep Forecasting. 65
4.2 Proposed DBN model for features learning. 66
4.3 Weights distribution of pretrained DBN 68
4.4 Weights representations of pretrained DBN 69
4.5 Weights representations after fine tuning 70
4.6 ANN model trained on features extracted from DBN. 72
4.7 Close-loop ANN model for forecasting future samples. 72
4.8 Learnt features extracted from hidden layers of DBN 73
4.9 Fifty samples of 4 steps ahead forecasted Temperature. 75
4.10 Error Histogram for next 1 hour, 4-steps ahead forecast. 76
4.11 Regression plot for next 1 hour, 4-step ahead forecast. 76

5.1 Flow of BCI System 81
5.2 Flow of Research Methodology 84
5.3 Emotive EEG Neuroheadset set Electrodes position 85
5.4 Eye Open and Eye Close states 85
5.5 Eight level DWT decompositions of EEG signals 87
5.6 Presentation of Four-level DWT decomposition tree 87
5.7 SAE 2 model for classification 89
5.8 DBN model for classification 90
5.9 Statistical Range of Electrode signals for eye open state 91
5.10 Statistical Range of Electrode signals for eye closed state 91

6.1 Outliers detection in Rain dataset. 97
6.2 Filtered Rainfall data. 98
6.3 Convolutional Neural Network for rain prediction. 103

List of Tables

2.1 Classification error on MINIST dataset 23

3.1 Optimal Parameter settings for pretraining. 34

3.2 Performance measures for predicting Lorenz and Mackey-Glass time series 38

3.3 Performance measures of Model I, II and III 38

3.4 Feature selection using Mutual Information 50

3.5 RMSE and MSE on Training and Test sets for Prediction of Metrological Parameters. 57

3.6 Error Comparison with other traditional approaches for Temperature prediction. 60

3.7 Error Comparison with other traditional approaches for Humidity prediction. 60

3.8 Error Comparison with other traditional approaches for Pressure prediction. 60

4.1 Optimal Parameter settings for pretraining of DBN model. 67

4.2 Error and Mean Approximation of pretraining. 67

4.3 Error rate Comparisons with Conventional Models 77

4.4 MSE for top three best multistep forecasters 78

5.1 Parameter Settings for SAE models 88

5.2 Parameter Settings for Pretraining of DBN model 90
5.3 Performance Metrics of Trained models on Test set 92
5.4 Comparison with Previous Research 93

6.1 Mean Square Error on Filtered Rain data. 98
6.2 Performance measures of proposed deep architectures for Eight step ahead rainfall forecasting. 102

Chapter 1

Introduction

Over the last few decades, the research community has been seen widely attracted towards the time series analysis, modelling, prediction and forecasting.

Anticipating the future values of an observed time series phenomena plays a significant role in almost every field of science, medicine and engineering for example, meteorology , telecommunication, control systems, biomedical signals, economics, business intelligence, finance and so on. Predicting future is one of the most challenging tasks in science. Anticipating adequate predictors and indicators from historical data requires statistical and computational methods for deducing dependencies between past and future values of observed samples and techniques to cop up with longer horizons [1].

The identification of past patterns and their extrapolation into the future is to treat them as comprising of four elements: seasonality, trend, cyclicality and randomness (error terms). The first three elements, seasonality, trends and cyclicality can be extrapolated to prepare more accurate forecast, while the randomness cannot be predicted. However, randomness indicates the extent of variation between the actual and predicted results. Randomness is also called residuals. Randomness can help us determine the extent of uncertainty in our future predictions. This does not mean there is error free forecasts. This actually means that explicit systematic forecasting approach can provide substantial benefits when used properly as all types and forms of forecasting techniques are made available within the existing data.

Multi-step ahead forecasting tasks are more difficult and challenging as compared to one-step ahead forecasting. Due to the fact that multi-step ahead forecasting

encounters several additional complexities such as accumulation of errors, increased uncertainty and consequently reduced accuracy [2],[3],[4]. Additionally it is volatile, thus making the process of forecasting cumbersome. Therefore, a successful forecasting model must be able to capture long term dependencies from the past chaotic data.

Time series may have variety of features. Some may posses seasonality, some exhibit trend (exponential or linear) and some are trend-less, just fluctuating around some level. In general, some preprocessing needs to be done to handle these features.

In the forecasting community, machine learning approaches are usually known as black-box or data-driven models and have proved themselves as strong contenders to traditional statistical models [5].

Over the years, enormous research has been done and various time series forecasting models have been developed in literature. In order to scrutinize and to further deal with time series forecasting for the ever increasing amount of data still much remains to be done to further investigate the role and potential benefits of different machine learning approaches. We have particularly focused on four architectures related with the family of deep learning models in our thesis. However, the complication and difficulty in assessing the exact nature of a time series, it is often considerably challenging to generate appropriate forecasts. In this thesis, we have attempted with various different kinds of time series related with meteorology, traffic data and biomedical signals.

With the available time series data, historical values of a time series can be treated as lagged terms. The selected lagged samples can play a significant role for time series analysis in terms of an input to the designed model. The appropriate selection of lagged terms can further encourage the forecasting model to accomplish better prediction accuracy. The delays are an attempt to add a temporal dimension to the network which is not present in Recurrent Neural Networks or Multi-Layer Perceptrons with a sliding window. The combination of past inputs with present inputs make the Time Delay Neural Networks (TDNN) approach unique.

A key feature for TDNN's are the ability to express a relation between inputs in time. This relation can be the result of a feature detector and is used within the TDNN to recognize patterns between the delayed inputs.

The upcoming sections describe in details the concept of time series forecasting and the ways through which it can be computed successfully along with the available constraints. Subsequently, the state of the art related with time series forecasting is also discussed later in this chapter.

1.1 Time Series Forecasting

Predicting future observations of a collected sequence of historical observations is called time series forecasting. Massive amount of research has already been carried out by researchers over many years for the development of models to improve the time series forecasting accuracy. The major aim of time series modelling scrupulously examine the past observation of time series to develop an appropriate model which elucidate the inherent behaviour and pattern existing in time series. This postulated model is further utilized as a forecaster to generate the future values. It is clear that accurate forecasting depends on appropriate model fitting. Thus, proper care needs to be taken to fit an adequate model to the underlying time series. Since many years, the ANN are playing significant part in the domain of time series forecasting and are successfully applied in several different areas [8,13,20]. The ability of nonlinear modelling, beyond any presumptions about statistical distribution of given data is a magnificent quality of ANN. Consequently, ANNs are data driven and self adaptive by nature.

Forecasts are essential, considering the fact that they guide decisions in many areas of scientific, industrial and economic activity such as in meteorology, telecommunication and finance. Time series data can exhibit many different patterns over time, out of which two can be generally identified, the trend and seasonal variations. The trend pattern can be described as long-run increase or decrease in the altitude of time series. It can also exhibits fluctuations that or not of fixed period, which can be termed as nonlinear trend. The seasonal pattern can be described as periodic oscillations determined by seasonal factors that are of fixed intervals.

From statistical perspective, the observed value $y(t)$ at time t can be considered as a realization of an underlying random variable. In the same way, a time series $[y(t), y(t+1).....y(t+N)]$ can be presented as final realization of a stochastic process which can be described as family of random variables indexed over time. Usually, the forecasts are estimated on the basis of some observations, for example,

historical data. If we assume that information given by X, then the conditional distribution can be specified as in equation 1.1.

$$f(y(t)|Y(t-1)) \tag{1.1}$$

However, modelling the conditional distribution is not an easy task. Thus , the conditional mean and conditional variance are attempted to be estimated as defined below.

$$\mathbf{E}[y(t)|X] \tag{1.2}$$

Time series containing records of single entity is called univariate. However, if records of multiple variables exist than it is known as multivariate.

The necessity of suitable input variable and lag selection becomes principally important for any forecasting model. As the input feature vector needs to capture all characteristics of underlying time series , containing the elements of deterministic or stochastic trends, cycles and seasonality. In general, for univariate time series, the analysis are conducted on the basis of using only lagged sample terms of the dependable variable. On the other hand, the multivariate modelling may opt the integration of explanatory dummy-variables [6].

1.2 Towards Machine learning, Neural Networks and Deep learning

In last few decades, the use of machine learning has spread rapidly beyond the limitations of computer science field. Machine learning is extensive and so pervasive today that one probably uses it dozens of times a day without knowing it. Many researchers also believe that it is the best way to make progress towards human-level AI. In the past decade, machine learning has given us self-driving cars, practical speech recognition, face recognition, effective web search, and a vastly improved understanding of the human genome.

Machine learning is a type of Artificial Intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. The existence of machine learning emerged from the study of pattern recognition and computational learning theory in artificial intelligence. There are actually thousands of machine learning algorithms available and more than hundreds are published each year. Now, here comes a question for one's application, " Which one could be the best selection?". In order to not get lost in the gigantic space, the key for model selection is to realize that it consists of the fusion of three components as presented below.

$$\text{Learning} = \text{Representation} + \text{Evaluation} + \text{Optimization}$$

Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data. In supervised learning applications in machine learning and statistical learning theory, generalization error (also known as the out-of-sample error) is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.

Generalization usually refers to machine learning model's ability to perform well on unseen data rather than only on the training data. If a model is over-trained generally known as overfitted, it will not generalize well. This indicates that generalization being the goal has interesting consequence for machine learning. A machine learning algorithm is used to fit a model data by performing a training procedure. Once training is over, the model is executed upon new data. Now, this is the exact point where problems can start appearing. As mentioned earlier, if a model is over-trained on training dataset, it will be able to identify all the relevant

information in the training set. Although, it will fail miserably when presented with new data. On the whole model is incapable of generalizing.

In supervised machine learning algorithms generalization error can be minimized by avoiding the overfitting and underfitting in learning algorithm, which is usually considered as Bias-variance tradeoff. The deep learning models as explained in the next chapters are basically trained into two segments. Firstly, as unsupervised models and than subsequently fine tuning is done in supervised manner. Number of studies in the deep learning algorithm prove that the unsupervised pre-training works as an unusual form of data-dependent regularizer. Thus, decreasing the variance and proposing the bias towards configurations of parameter space that are useful for further supervised training. In this thesis, specifically in later chapters, we show the performance results of our models on both training and test sets. The error on the test dataset is much more lower than the training error due to unsupervised pretrainings.

Subsequent to overfitting, the next threat to machine learning algorithm is the "curse of dimensionality". Deep learning resolves it. The curse of dimensionality is a phenomena that happens when the dimensionality of the data increases. The more dimensions are added to the data, the more difficult it becomes to find the pattern. What overcomes the curse is a better modelling of data, particularly modelling it in a lower dimensional space, where the relative position of points in that space convey information about mutual information. Deep learning has overcome this curse by its inherent nature. Deep learning does not suffer from the same consequences as other machine learning algorithms face in high dimensionality applications.

Feature engineering is typically where most of the effort in a machine learning model development goes. Manifestly, one of the holy grails of machine learning is to automate more and more of the feature engineering process. Various unsupervised machine learning algorithms contemplate on discovering better representations from the inputs provided for training. Representation learning algorithms and manifold learning algorithms also attempt to do so under the constraint that the learned representation is in the low dimensional space.

1.3 Motivation and Aim

Time series forecasting remains a challenging problem despite the fact that there is substantial improvement in machine learning approaches. This usually happens due to some factors like, different time series may have different flattering behaviour. In real world time series data, the discriminative patterns residing in the time series are often distorted by random noise and effected by high frequency perturbations.

The major aim of this thesis is to contribute to the study and expansion of time series prediction and multistep ahead forecasting method based on deep learning algorithms. The journey of my research started in 2014. Time series forecasting using deep learning models was still in infancy as compared to other research areas statistical or machinelearning models for time series forecasting. The availabity of relevant literature was scarce. However, there was alot of work on sequential supervised learning in the scope of deep learning by using the architectures, such as, RNNs and Lstms. If we further explore it there is a little difference between sequentiql data and time series data. The sequential data has an order irrelavant of time stamp. On the other hand the time series data has fixed intervals along with the ordered structure. Consequently, this gave us direction to further investigate this field and study that which kind of representations can be extracted from these deep learning archirtectures over time series data.

Deep learning is a category of machine learning models. Deep Learning Architectures are able to extract high level abstractions from input distribution of data by by means of multiple processing layers, composed of multiple linear and non-linear transformations. One of the primary characteristic of deep learning is to replace hand crafted feature set extracted from input data, with efficient algorithms for unsupervised feature learning and hierarchical feature extraction.

The depth is actually inspired by both the biological brain system and theory of circuit complexity, which influence that deep architectures can be exponentially efficient than the shallow ones.

Several variational forms of Deep learning architecture exist in literature for example, Deep belief networks, deep neural networks, deep convolution neural networks, recurrent neural networks and stacked Autoencoders. Present literature suggests that these architectures are being applied and has produced state of the art

results on various problems in major fields like computer vision, automatic speech recognition, natural language processing, audio recognition and bio-informatics.

The standard training of deeper models through gradient backpropagation appeared to be difficult until Hinton gave a breakthrough in 2006. The standard training strategies attempts to allocate the parameters in the region of parameter space that generalize poorly. This has been shown practically in number of studies [7].

1.4 State of the art

The forecasting domain for a long time has been influenced by traditional linear statistical methods for example, ARIMA models or Box-Jenkins [8–12]. However, these methods are totally inappropriate if the underlying mechanism is nonlinear. The real life process is mostly nonlinear which brings deficiency in traditional mathematical model. Later in 70's and early 80's, after the realization that linear models may not adapt to the real life process several useful nonlinear models were developed [6][7].

We drive the literature review starting from machine learning models for time series forecasting towards neural networks and its variants. Moreover, we move towards the time series analysis and forecasting by recent deep learning architectures. Literature survey apprise that since the last two decades, the machine learning algorithms have established themselves as serious contenders to traditional statistical models.

The research in [13] presented large scale comparison study for major machine learning models. The study is based on the benchmark dataset of Monthly M3 time series competition data. Their investigation revealed significant differences between different models such as, multilayer perceptron, radial basis functions, k-nearest neighbour regression, Bayesian Neural Network, kernel regression, CART regression trees, Gaussian process and support vector regression. In their research, the authors further provide precise ranking for above mentioned machine learning models. However, it was also identified that the achieved ranking is some how specific to business type time series. Their study reveals that the MLP and then the GP as the two best models for time series forecasting.

There are numerous comparison studies that compare neural networks with traditional linear techniques for forecasting. For example,

In [14], a new forecasting strategy known as rectify is proposed. The strategy combines the recursive and direct forecasting approaches. In order to investigate the performance of rectify strategy, real word time series from the M3 and NN5 forecasting competitions are used in addition with simulated liner and nonlinear time series.

Literature indicates that several approaches that deals with the complex problem of multi-step forecasting have been proposed. The authors in [4] provided detailed review on existing strategies. The most noticeable findings in their research are that Multiple-Output approaches are constantly better than Single-Output approaches, the positive impact of deseasonalization on the performance and input selection in conjunction with deseasonalization is more effective.

Moving towards the multi step ahead forecasting problem, according to the authors in [4], five different strategies have been proposed to solve the issue of H-step ahead forecasting task. For example in the Recursive approach, one step ahead forecasting model is iterated H times to obtain the H forecasts [3],[15],[16],[17]. The Direct strategy implements set of H forecasting models unlike the Recursive; Each model producing the forecast for the ith value (i $\in$ {1,....H}) [3],[15],[16],[17]. The researchers in [18] proposed a combination of the two previously mentioned strategies called as DirRec. In order to sustain the stochastic dependency of time series, Multiple-Input Multiple-Output (MIMO) strategy is acquainted and analysed [17],[19],[20]. MIMO produces the vector of future values in a single step. DIRMO is the fifth strategy which preserves the captivating aspects of both Direct and MIMO [21]. However, a lot of research have been also conducted for comparative analysis of these strategies [4, 22]. These comparisons are conducted by configuring different forecasting models, for instance, Artificial Neural Networks, Multiple Linear Regression, Hidden Markov Models and Nearest Neighbours.

Neural networks are empirically evidenced as more powerful also in examining and understanding the dynamics of financial time series in comparison to traditional statistical models [23–27].

There research in the machine learning literature has paid very little attention on the forecasting strategies in the scope of time series modelling. The major focus is given on the prediction accuracy and computational complexity. The research

conducted in [13] presents comprehensive comparison of various machine learning algorithms implemented for time series forecasting.

Literature suggests that several different ANN models have been proposed since 1980s. Perhaps, the most influential ANN models particularly for time series forecasting are MLP, RNN, TDNN, Kohnen's Self Organizing networks. Infact, Artificial neural networks are the most common machine learning algorithms applied on various time series related with different domains [28–32]. Apart from this, Several variants of these models have been developed in literature [33–35].

Recently, the time series forecasting task has been also computed by using a particular type of ANN known as Extreme Learning Machines (ELM). The major feature of ELM is learning without iterative training. This makes the learning process faster when compared to the traditional algorithms for the training of neural networks For the recent advances with ELM, see [36–44].

A number of other machine learning algorithms have received little attention in the forecasting literature. In the territory of the recent machine learning literature particularly the ANN, the attention has diverted towards Deep learning and its incredible applications. Being particular, by keeping our focus in the realm of time series forecasting and prediction, we shed some light by presenting some research related with time series forecasting with deep learning methods. As aforementioned in previous section, various different architectures comes under the umbrella of "Deep Learning models". The initial research advocates that DBN models are efficient at the classification and prediction tasks but, it actually lacks the efficacy to model temporal sequences. The authors in [45] have reported that the recurrent neural networks performed drastically better on energy load forecasting using dataset from kaggle competition. They further argued that the greedy layer-wise trained feed forward neural networks with stacked autoencoders obtained discouraging results with no significant performance gain but added complexity. As it is well known fact that feed forward networks are deficient in capturing temporal dynamics. Thus, unable to access the past terms while modelling the underlying structure.

On the contrary, the Stacked Autoencoders are also deployed in [46] in order to learn feature representations for weather forecasting. The empirical evaluations are further compared between models using raw features and models using learned representations as features. The obtained results in the above mentioned study prove that Deep Neural Network (DNN) is capable enough to provide better feature space

for highly varying and non-stationary data like weather data series of temperature, pressure and wind speed. Related study has been provided in [47] for short term wind prediction and in [48] for load forecasts.

A predictive model has been proposed for time series data in [49] by using a DBN with RBM. Additionally, the performance of the proposed model is evaluated on data of CATS benchmark [50, 51] and chaotic time series. According to the experimental outcomes, it was confirmed that the proposed prediction model, DBN with RBM using pretraining and fine-tuning learning algorithm and PSO structure decision, performed better than traditional models although it is unable to beat the best of IJCNN 2004 competition model. The research in [52] presents a novel hybrid model with discriminative and generative components for spatio-temporal inference about weather. Furthermore, a data driven kernel is implemented that forms the predictions according to physical laws.

A detailed review of unsupervised feature learning and deep learning for time series modelling has been conferred in [53]. The article presents the detailed review on time series analysis and temporal sequence modelling using deep architectures. However, according to our observation, the study was found to be deficit, as far as the overview regarding time series forecasting with deeper models is concerned. The CRBM was introduced in the family of Deep Learning by applying it to capture the activity related to human motion [54]. Similarly, the other variants od RBM, for example, Temporal RBM [55, 56] and Gated RBM [57, 58] have also been introduced. With the exception to these models, another deep architecture producing outstanding state-of-the-art research is Convolutional Neural network (CNN). These models are of high interest specifically for image data or high dimensional time series data. Apart from being stand alone, convolution has also been applied as convolutional RBM [59, 60] and Convolutional AutoEncoders [61–63].

Although, a lot of research has been carried out for sequence learning in the initial era of deep learning and other complex problems but very little attention was paid towards the time series forecasting tasks using deep learning architectures. However, the area of time series forecasting using Deep Learning Architectures can be seen as growing rapidly, particularly in the year 2016 and 2017.

In [64] , the authors implement Deep Learning Networks for stock market analysis and prediction. Similarly, Deep networks are used for predicting the direction of change in foreign exchange rates [65]. The authors proposed a new method in

[66] which integrates dictionary learning in sparse coding into a stacked autoencoder network and extracts features automatically from raw vibration data. Similar studies can be found in [67–73]. The research in [74] examines the role of memorization in deep networks.

1.5 Main Contributions

Primary goal of this dissertation is to develop deep learning models that can automatically learn the representations and inherent structure of time series data. The work presented in this thesis is an experimental study for the behaviour and performance of different Deep learning architectures on temporal data. The major thesis contributions are as follows:

- Four different types of deep learning architectures are considered for the prediction and classification of time series data, which include DBN-DNN, CNN, LSTMs and SAE.
- The suitable and adequate architectural topology for developing hierarchical structure for deep networks is suggested in terms of hidden layers dimensions. Moreover, this has been followed in our work as standard approach for selecting the size of hidden layers.
- A hybrid approach for multistep time series forecasting is proposed. The proposed approach lies in the category of Recursive forecasting strategy. The performance is further evaluated by providing the comparison with other traditional methods.
- Multistep forecasting is computed through DBN model on rainfall data. This follows the direct forecasting strategy.

1.6 Publications

- Sanam, Narejo ; Eros, Pasero. An Application of Internet Traffic Prediction with Deep Neural Network. In: WIRN 2016, 26th Italian Workshop on Neural Networks, Salerno(IT), 18-20 May 2016.
- Khurshid, Aliev; Sanam, Narejo; Eros, Pasero; Jamshid, Inoyatkhodjaev, (2016). A Predictive Model of Artificial Neural Network for Fuel Consumption in Engine Control System. In: WIRN 2016 26th Italian Workshop on Neural Networks, Salerno (IT), May 18-20, 2016.

- Sanam, Narejo ; Eros, Pasero. (2016). Time Series Forecasting for Outdoor Temperature using Nonlinear Autoregressive Neural Network Models. In: JOURNAL OF THEORETICAL AND APPLIED INFORMATION TECHNOLOGY, vol. 94 NO. 2, pp. 451-463. - ISSN 1992-8645
- Sanam, Narejo; Eros, Pasero. (2016). A Hybrid Approach for Time Series Forecasting Using Deep Learning and Nonlinear Autoregressive Neural Networks. In: INTELLI 2016 : The Fifth International Conference on Intelligent Systems and Applications, Barcelona (Spain), November 13, 2016 to November 17, 2016. pp. 69-75
- Sanam, Narejo; Eros, Pasero; Farzana, Kulsoom;(2016). EEG Based Eye State Classification using Deep Belief Network and Stacked AutoEncoder. In: INTERNATIONAL JOURNAL OF ELECTRICAL AND COMPUTER ENGINEERING, vol. 6. No. 6. - ISSN 2088-870
- Sanam, Narejo; Eros, Pasero; (2017). MeteoNowcasting using Deep Learning Architecture. In: INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, Vol. 8. No. 8.
- khurshid, Aliev; Sanam, Narejo; Eros, Pasero; Alim, Pulatov; Internet of Plants Application for Smart Agriculture. Submitted to: COMPUTERS AND ELECTRONICS IN AGRICULTURE.

1.7 Organization of thesis

This thesis presents a collection of the work done on time series forecasting using deep learning models. Particularly, bringing up some novelty in earlier deep learning models like, DBN and SAE to work with temporal sequences and are further compared with other state of the art models like RNNs and LSTMs. CNN is also introduced in Chapter 6 for rainfall data. Additionally, to produce accurate estimations not only for one 1-step ahead predictions but also multi-steps ahead. The two variations of forecasting strategies, recursive and direct approach both are considered for multi-step ahead forecasting. Necessary data transformations are applied prior to implementing the forecasting models. The effectiveness of our proposed approach was further tested on two benchmark time series. A brief description of each chapter is presented here.

- Chapter 1: This chapter works as initialization. This chapter introduces the basic terminology and primitive concepts to get familiar with time series, its analysis, associated challenges while examining underlying time series and forecasting in association with machine learning. Additionally, it also provides a journey over the deep learning architectures used for the analysis, prediction and forecast of time series datasets.

- Chapter 2: Basic theory regarding Deep learning architectures, particularly those which are implemented in our research has been described here. This chapter provides the conceptual background for deep learning architectures.

- Chapter 3: This chapter gives a description of implemented models. The significant parameter settings, the dimensions for hidden layers and training strategies are explained in details. The chapter also provides the next step forecasting for internet traffic data and meteorological data, such as, temperature, pressure and humidity for METEO station. Moreover, the performance of developed models is compared with other traditional approaches available in literature.

- Chapter 4: In this chapter, we propose a novel approach for multi-step ahead forecasting of time series. It is a hybrid approach using deep belief networks and Nonlinear Autoregressive with eXogenous (eXternal) inputs neural model. We use the proposed model for temperature forecasting. The model improved over few approaches except the Nonlinear Autoregressive Moving Average NARMAX neural model. Thus, it gave us reasoning to further extend this model based upon NARMAX model.

- Chapter 5: This chapter presents the use of deep learning techniques to classify the eye state: whether open or closed from EEG signals and further demonstrates the improved performance accuracy over previous related state of the art research. We used SAE and DBN for the classification tasks and compared it with the traditional models available in literature.

- Chapter 6: This chapter presents another extended approach for multi-step ahead forecasting of rainfall time series. The proposed direct approach provides better performance as compared to other baseline methods developed for comparison on the rainfall data set.

- Chapter 7: Final concluding remarks of the conducted research, its limitations and future directions are discussed in this chapter.

Chapter 2

Deep Learning Architectures (Theoretical Concepts)

2.1 Introduction

In this chapter, the brief theoretical background of deep learning algorithms is presented in order to understand the concepts and methods of our research that are implemented in later chapters.

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data. For example, neural networks with many hidden layers forms a deep architecture composed of multiple levels of non linear operations. When the neural network is composed on more than one or two hidden layers, contingent upon that situation the higher layers compose abstractions on the top of previous layers. Howbeit, before the breakthrough (Hinton, 2006) increasing the number of hidden layers accelerates to two very well known issues, i.e, the problem of vanishing gradients and overfitting. Searching the parameter space of deep architectures is a difficult task. In the next sections, we will discuss how deep learning algorithm with many hidden layers takes turn on the earlier mentioned problems by considering the ideas given by the pioneers of this field.

The choice of data representation is highly effective and influential on the performance of any machine learning algorithm or model. Deep learning aims to learn good representations from low level features. This automatic learning of features at hierarchical levels allow a network model to learn complex functions and mappings

from the input distribution, without relying on hand engineered or human crafted features.

The researcher in [75] argued that deep learning earns kind of distributed representations as features and are invariant. These learned representations might be helpful in dealing the curse of dimensionality and the limitations of local generalization. These kind of distributed representations of data are also inferred from our implemented predictive models as presented in Chapter 3 and Chapter 4. In the distributed representation, these representations are not mutually exclusive and may be statistically dependent or independent.

Initially, Hinton et al proposed the concept of Deep Belief Networks, with a learning algorithm that greedily trains one layer at a time in an unsupervised manner using Restricted Boltzmann Machine [73]. Later in a short time, relevant algorithms based on autoencoders were suggested [17 153]. Other algorithm with deep architectures are also proposed and are attaining state of the art performance.

In the very beginning, the evolved deep learning approach was applied primitively on MINIST digit set as an image classification problem [76], [77]. Currently, the deep learning approaches are being applied on object recognition and scene recognition. Breakthrough results are obtained by deep learning and representation learning in the area of speech recognition [78],[79],[80],[81]. Deep learning algorithms are also practiced on music generation and achieved state of the art performance in polyphonic transcription [82] and competition on audio tagging [83]. In the field of Natural language processing (NLP) the study for symbolic data were introduced in [84] and were firstly developed in the context of modelling for statistical language in [85],[86]. Representation learning has been successfully implemented and has surpassed the state of the art sentiment analysis [87]. [88], [89].

2.2 Deep Belief Network (DBN)

DBNs are intended to be one of the foremost non-Convolutional models to successfully admit training of deep architectures. Earlier than the preface of DBN, deep models were hard to optimize [75]. DBN is basically a generative model with various layers consisting of latent and visible nodes. The latent variables are typically the binary units, while the visible units have attainability to be binary or real. The layered

structure in DBN can be formed by stacking RBMs which are used to initialize the network in the region of parameter space that finds good minima of the supervised objective. In other words, we can say that the RBM is a building block of DBN.

A graphical model of DBN is stacked up with layers of Restricted Boltzmann Machines (RBMs) as presented in figure 2.1. The DBN model is trained by training RBM layers using contrastive divergence or stochastic maximum likelihood. The parameters of RBM then designate the parameters of first layer of the DBN. The second RBM is trained to model the distribution defined by sampling the hidden units of the first RBM whose visible layer is also working as an input layer as well. This procedure can be repeated as desired, to add as many layers to DBN.

The whole procedure of training is divided as:

1. Constructively adding the layers one by one and each layer is trained separately in an unsupervised manner.
2. The next added layer takes the input from the output of the previously trained layer. This indicates each layer takes the trained representation as an input.
3. The last layer is added with some cost function and targets to predict the final output.
4. Finally, the whole constructed network is pretrained with supervised strategy which is known as a fine tuning step.

This indicates that once meaningful or significant representations are found at each level, these abstractions are used to initialize and train a deep NN in a supervised manner. The unsupervised pretraining constraints the network parameters in the better region where the solution is allowed. Subsequently, the unsupervised pre-training is also identified as data-dependent regularizer as it brings the test error down as compared to the training error rate.

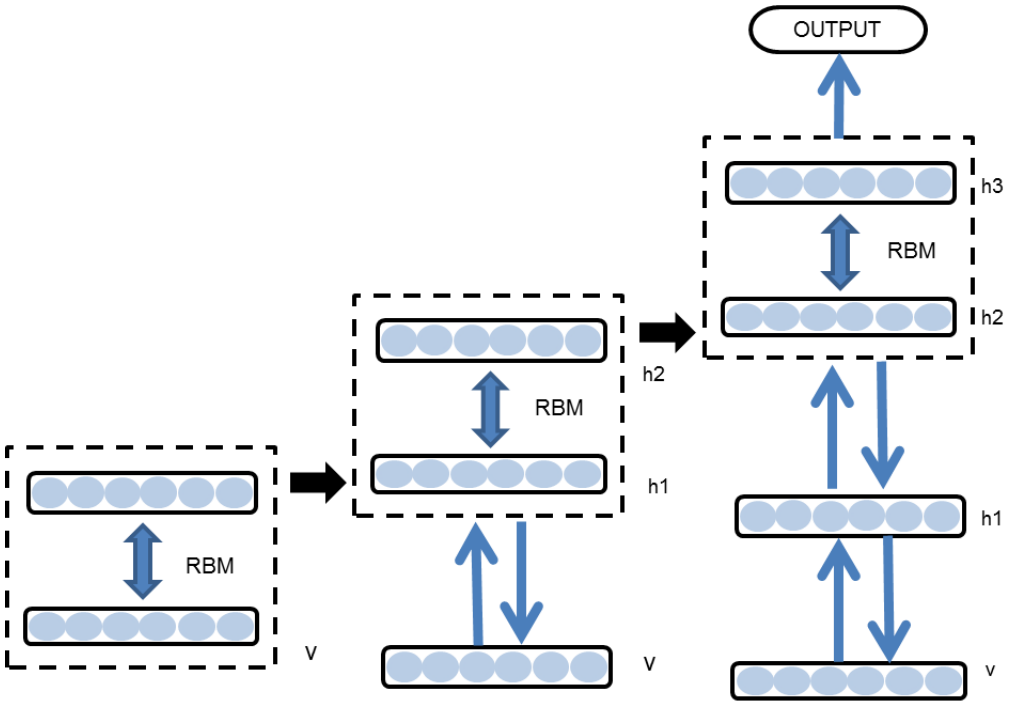

Fig. 2.1 A stack of Restricted Boltzmann Machines (RBMs), greedily trained as Deep Belief Network (DBN).

2.2.1 Restricted Boltzmann Machine (RBM)

RBMs are some of the most common building blocks of deep probabilistic models. RBM is energy based, probabilistic graphical model. RBM relies on two layer structure comprising on visible and hidden nodes as shown in figure 2.2.The visible units constitute the first layer and correspond to the components of an observation where as the hidden units model dependencies between the components of observations. Probability distribution in energy based models is given by an energy function, given as follows. Where Z is a normalizing factor, v is observed and h is representing a node in hidden layer.

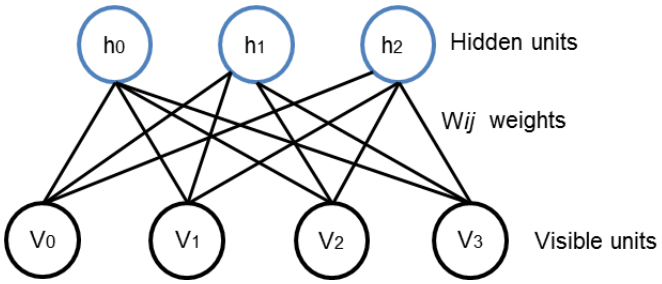

Fig. 2.2 Graphical structure of RBM with visible layer and hidden layer

$$p(x) = \sum_{h} p(v,h) = \sum_{h} \frac{e^{-E(v,h)}}{Z} \tag{2.1}$$

In similar case, the free energy from the physics law, can be expressed as follows

$$F(x) = -log \sum_{h} \frac{e^{E(v,h)}}{Z} \tag{2.2}$$

The energy function of RBM is defined as

$$E(v,h) = -h'Wv - b'v - c'h \tag{2.3}$$

RBM parameters can be denoted as θ = {W,b,c}, where W is the weight matrix, b is the bias vector for visible units v and c is the bias vector for hidden units h.Because of the specific structure of RBMs, visible and hidden units are conditionally independent given one-another. Using this property, we can write:

$$p(h|v) = \prod_{i=1}^{n} p(h_i|v) \quad and \quad p(v|h) = \prod_{j=1}^{m} p(v_j|h). \tag{2.4}$$

With parameter θ the gradient of log likelihood given over a single training example v is further computed as

$$\frac{\partial ln(\theta|v)}{\partial \theta} = -\sum_{h} p(h|v) \frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta} \tag{2.5}$$

The above mentioned equation has two terms. The first term is conditional distribution. The second term is problematic because the computation of the expectation over the joint distribution is untraceable. The equation is basically a difference between

two expectations:the expected values of the energy function under the model distribution and under the conditional distribution of hidden variables given the training set. The computational complexity of calculating above expression is exponential in the number of variables in the MRF. Therefore, expectations are approximated from the distributions based on Markov chain Monte Carlo (MCMC) technique i.e. Gibbs sampling. Then the binary states of the hidden units are all computed in parallel using equation (2.6). Once binary states are chosen for the hidden units, a "reconstruction" is achieved by setting each vj to 1 with a probability given in equation (2.7).

$$p(h_i = 1|v) = sigmoid(\sum_{j=1}^{m} w_{ij}v_j + c_i) \tag{2.6}$$

$$p(v_j = 1|h) = sigmoid(\sum_{i=1}^{n} w_{ij}h_i + b_j) \tag{2.7}$$

The weight wij can be updated using difference between two measured data dependent and model dependent expectations as expressed in equation (2.8). Where ε is a learning rate.

$$\Delta wij = \varepsilon(< vjhi >_{data} - < vjhi >_{recon}) \tag{2.8}$$

It is important to make the hidden states binary instead of assigning probabilities at this stage. Because, if the probabilities are used, each hidden unit will convey a real value to visible units instead of binary state. This further breach the fact that hidden unit can only communicate one bit either 0 or 1, which actually influences the abstracted model to act as a regularizer. However, on the final step of contrastive Divergence for hidden unit updates, the probabilities are used.

2.2.2 RBM for Real Data

We have developed and trained the binary RBMs in our DBN implementations as predictive models. Although our continuous valued input was scaled in the interval of (0,1). In this manner, each input was considered as a probability for binary random variables to take the value 1 and this performed quite well. However, there is a limitation in Gaussian and exponential hidden units, i.e, the mean field approximation using a Gaussian unit gives rise to purely linear transformation. Additionally, in a DBN containing only Gaussian units one would only be able to model Gaussian data.

In order to show the efficacy of unsupervised pretraining a table is taken from the experiments conducted in [77]. It shows the classification error on MiNIST dataset which includes performance comparison between shallow, supervised and unsupervised pretraining. Although, in all of the presented approaches fine tuning was performed by adding the logistic regression layer and each model was trained by stochastic gradient descent. For detailed explanation on architecture and parameter settings see [77].

The presented error results in Table 2.1 suggest that unsupervised pretraining is capable to produce much better results than shallow network models and supervised training approaches. Furthermore, the performance of pretrained autoencoder model is comparable to DBN.

Table 2.1 Classification error on MINIST dataset

Models	**Training**	**Validation**	**Test**
Deep neural network, no pretraining	0.004	2.1%	2.4%
Shallow neural network, no pretraining	0.004	1.8%	1.9%
Deep neural Network, supervised pretraining	0%	1.75%	2.0%
Deep autoencoder	0%	1.4%	1.4%
DBN, unsupervised pretraining	0%	1.3%	1.4%

2.3 Stacked AutoEncoder (SAE)

AutoEncoders are basically related with the category of representation learning technique which uses unsupervised pretraining to learn meaningful representations from the input distribution of data. Moreover, these are useful to reduce the dimensionality of any specific problem domain in order to facilitate the supervised learning task.

An alternative approach to deep learning is Stacked Autoencoder, where the hidden units generate exploitable numeric feature values. As demonstrated in [32, 33]. AutoEncoders have become the dominant focal point in the area of “deep architectures”.

An AutoEncoder is a neural network that consists of two parts: encoder and decoder. It is trained in such a way to reconstruct the given input at the output layer. The first part encoder is responsible to compute the hidden representations

h of input space v through some transfer function $f(v)$. The decoder takes these latents produced by encoders and further passes it through some non-linearity $g_{\theta'}(h)$ to generate the reconstruction output $\hat{v}$. The set of parameters θ and θ' are learnt simultaneously on the task of minimizing reconstruction error. $L(v,\hat{v})$ which is a measure of discrepancy between actual and reconstructed as expressed in equation 2.9. The Weight matrix of the decoding step W' is infact the transpose of weight matrix of the encoding part W in order to reduce the number of parameters to learn.

$$JAE = \sum L(v, g_{\theta'}(f_{\theta}(v))) \tag{2.9}$$

The AutoEncoder basically learns to map the input by learning the non-linear or some representations of input space in an unsupervised manner. The structure of an AutoEncoder with a single hidden layer is elaborated in Figure 2.3. The output layer in an AutoEncoder should be of the same size as the input layer, because the AutoEncoder is not classifying the labelled output, instead it is reproducing the input.

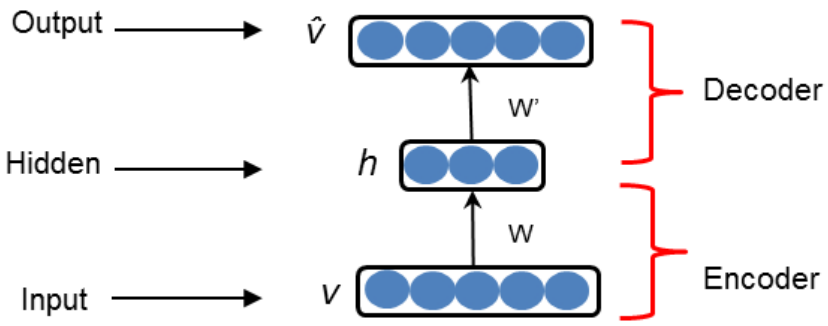

Fig. 2.3 Illustration of an AutoEncoder.

Each AutoEncoder is separately trained like an RBM in layer wise manner. Further, during the training procedure the reconstructions at the output layer are compared with the actual given input in order to encourage the neural network to produce the outputs as close to the input. Stacking the trained unsupervised AutoEncoder over another AutoEncoder and then training the whole network globally by adding an output layer places the Neural Network in the category of DLA. The formulating steps for constructive and iterative training of a stacked AutoEncoder are demonstrated in Figure 2.4.

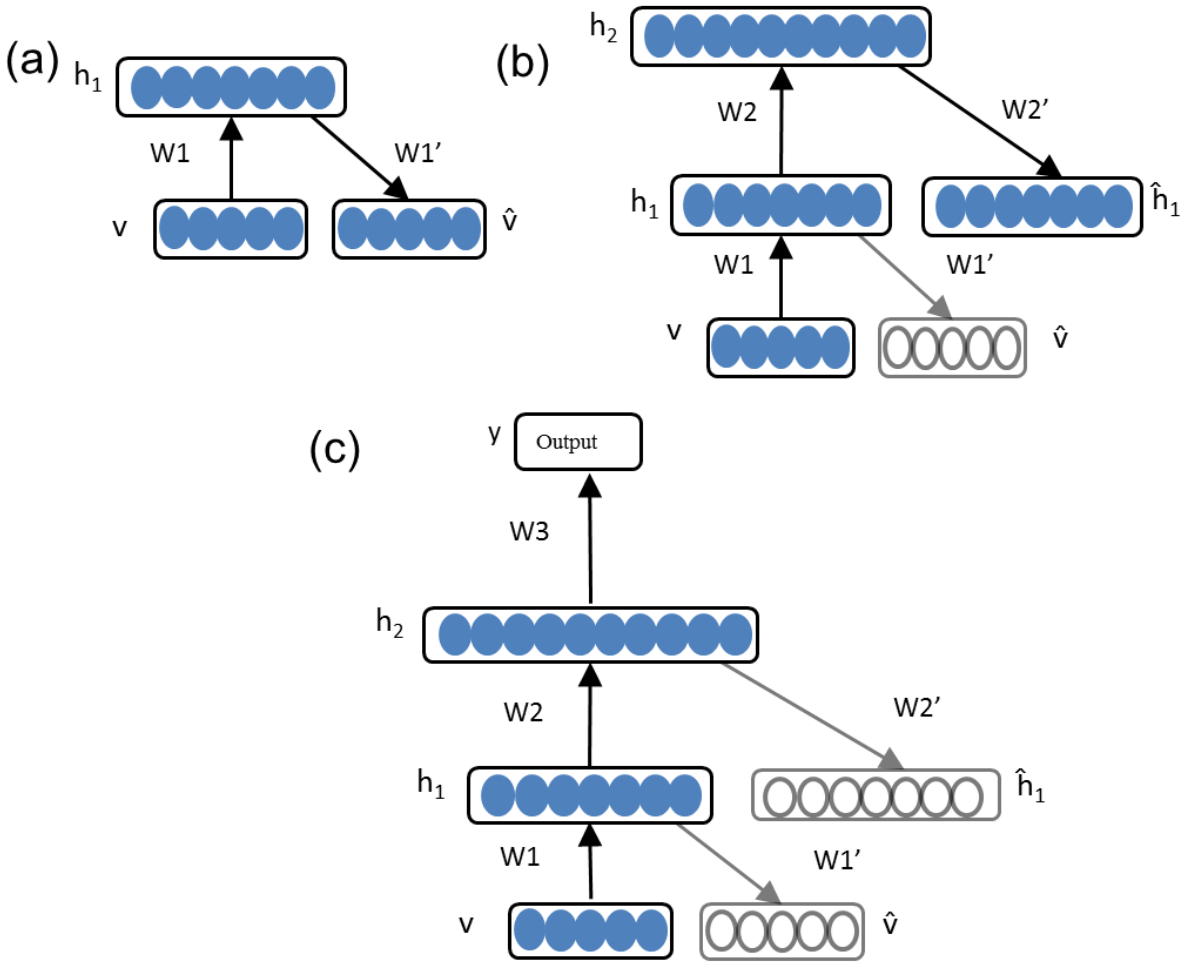

Fig. 2.4 Iterative training of stacked Autoencoder.

Considering the concept of feature or representation learning, one approach to earn useful features from the autoencoder is to limit the size or capacity of hidden layer. An autoencoder whose hidden layer dimension is smaller than the input layer dimension is known as under-complete This under-complete representation influences the autoencoder to seize the most salient features of the training data. On the other hand, if encoder and decoder is allowed to have too much capacity, the encoder performs the copying or mapping task with out any learning capability. If the hidden code has dimension greater than the input dimension, than it is known as overcomplete case. In order to resolve, the problem of overcomplete autoencoders without constraining the hidden layer capacity is to use Regularized AutoEncoders. A Regularized Autoencoder can be non linear and incomplete but still capable to learn some meaningful features from the input distribution of provided data.

Chapter 5 deals with the training, detailed implementation and further explanation for parameter selection of Regularized and Sparse autoencoders for a classification task. In a sparse autoencoder, training criteria involves a sparsity penalty $\Omega(h)$ on the coding hidden layer h in addition to the reconstruction error as expressed in equation 2.10.

$$JAE = \sum L(v, g_{\theta'}(f_{\theta}(v))) + \Omega(h) \tag{2.10}$$

2.4 Bells and whistles

The advantage of unsupervised pre-training versus random initialization was clearly demonstrated in several statistical comparisons [17, 50, 98, 99].

At the moment, in the latest research area of deep learning, plenty of techniques, appropriate recipes and tips are available for phenomenal training of deep generative models. There are some more effective and a proper guidance available for the selection of hyper-parameters settings recommended by Geoffery Hinton and Yoshua Bengio. We have empirically followed these provisions along with assumptions and understanding inferred from our perceptive experiments.

The study in [90] highlights few important points regarding the loss surface of multilayer networks and the behaviour of deep learning systems for optimization and generalization. It concludes that for larger networks, the local minima are almost equivalent which results in similar performance on test sets.It also shows that training and test error begin to be decorrelated as the network size is increased.

The deeper networks are tricky to train. The research in [74] explores the DNN optimization on noise vs real data and confirms the fact that DNNs trained with SGD-variants first use patterns, not just brute force memorization fitting real data. In our research work, We have not analysed the deep networks with noisy data. Although, we have done filtering to remove noisy fluctuations and interpolation for missing data prior to training as described in Chapter 3.

Chapter 3

Time Series Prediction using DBN

In this chapter, we present implementation and training details of our predictive DBN models. This chapter is divided into two major sections. The first task was to predict the internet traffic model with the help of deep learning approach. While running the experiments based upon the training of several architectural topologies of DBN we validate our choice of model selection on two benchmarks time series. Similarly, following the same approach we have done weather forecasting for Politecnico Di Torino. Part of the work described in this chapter has been presented and further published in [91, 92].

3.1 Hyper-parameter Settings

In machine learning, the term hyperparameter is used to discriminate from standard model parameters. In machine learning while fitting a model to data, a number of model parameters are needed to be learned from data, which is performed through model training. Moreover, there is another kind of parameters that cannot be directly learned from the legitimate training procedures. They are called hyperparameters. Hyperparameters are usually selected before the actual training process begins. The hyperparameters can be fixed by hand or tuned by an algorithm. It is better to adopt its value based on out of sample data, for example, cross-validation error, online error or out of sample data. However, in some learning algorithms like particularly unsupervised training algorithm hyperparameter. optimization is a bit problematic in this respect [93].

3.1.1 Learning rate

A default value of the initial learning rate ε_0 equals to 0.01 typically works well for standard Multi-layer ANN, albeit it is better not to rely on it without attempting some different trials. One useful tip for models to be trained is quoted in [93] as, "If there is only time to optimize one hyper-parameter using stochastic gradient descent, then learning rate is the hyper-parameter that is worth tuning".

3.1.2 Mini batch size

The impact of mini batch size B is almost computational, i.e, the larger value for B results in faster computation. This hyper parameter effects more on training time rather than the test performance. Although, it has been observed that it is better not to highly enlarged the size of mini batch when using stochastic gradient descent.

3.1.3 Number of iterations

As suggested in [93], this hyper parameters can be optimized almost for free using the principle of early stopping.early stopping rules work by splitting the original training set into a new training set and a validation set. The error on the validation set is used as a proxy for the generalization error in determining when overfitting has begun. Early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent. Such methods update the learner so as to make it better fit the training data with each iteration. It is an inexpensive way to avoid strong overfitting. However, it might be useful to turn off early stopping while analysing the effect of individual hyper-parameters.

3.1.4 Momentum

The standard Stochastic Gradient Descent (SGD) can cause very slow convergence particularly after the initial steep gains. Momentum is one method for pushing the objective more quickly along the shallow ravine. It is advocated in [94] to initiate the momentum with 0.5 and eventually increase the momentum to 0.9 once the error is close to saturation. Therefore, this policy of momentum has been

followed throughout of our research work. In [95], the researchers declared that both the initialization and momentum are critical. The well initialized models behave apparently worse in the absence of momentum or if the momentum is poorly tuned. On the other hand, if the network is poorly initialized but having appropriate momentum cannot be trained.

3.1.5 Regularization

In deep learning architectures, the regularization appears more like a tuning parameter to get a better test error rate. However, the absence of a regularization does not results in poor generalization error. Regularization can be explicit such as weight decay factor, dropout ratio and data augmentation or it can be implicit by model itself. The researchers also state that it is not necessarily true that regularizers are the fundamental reasons for generalization, as the networks are capable to perform well if all of the regularizing parameters are removed.

3.1.6 Hidden layer dimensions

The researchers have advocated in [7] that the reason for setting a large enough hidden layer size is due to the early stopping criteria and possibly other regularizers, for instance, weight decay, sparsity. Apart from this, the greedy layer wise unsupervised pretrainig also acts as data dependent regularizer.

In a comparitive study [96], authors found that using same size for all layers worked generally better or the same as using a decreasing size (pyramid like) or increasing size (upside down pyramid). They further argued that certainly this must be data dependent. However, in our research task the decreasing size structure worked far better than the other too. Consequently, this architectural topology was chosen as standard for further forecasting models.

The authors in [93] have argued that in most of the conducted experiments it was found that an overcomplete first hidden layer in which the dimensions are higher than the input layer dimensions, works better than the undercomplete one.

Due to the availability of high performance computing facilities and massive computational resources, the more productive and automated optimization of hy-

perparameter is possible through grid search or random search methods. We have applied both of the mentioned strategies in our experiments.

3.1.7 Computational resources

According to our major goal, we attempted for accurate time series forecasting and prediction with deeper architectures. Deeper architectures are little more exhaustive as far as the computational resources are concerned. Due to this, it took almost more than couple of days to well trained the deep herarichical model. However, our major concern was not about acceleration but accurate modeling of data. Albeit, the acceleration can be improved by utilizing the GPUs anf FPGAs.

The simulations were executed on 16 and 32 cores of High-Performance Computing (HPC) cluster of AMD Bulldozer CPU provided by HPC@POLITO. These clusters are characterized by many cores and processors, lots of memory, high-speed networking, and large data stores – all shared across many rack-mounted servers. User programs that run on a cluster are called jobs, and they are typically managed through a queueing system for optimal utilization of all available resources.

Apart from this, we also utilized the computational resources provided by NEURONICA laboratory and OPLON MIUR.

Computational resources were partly provided by HPC@POLITO,
(http://www.hpc.polito.it).

3.2 Internet Traffic prediction

The advance knowledge of future traffic load is helpful for network service providers to optimize the network resource and to recover the demand criteria. In this section, we present our work for time series prediction of internet traffic by developing the RBM based DBN model of deep learning. Additionally, we also demonstrate the significant strategies for the selection of optimal architecture for training the deeper hidden layers. This architecture selection is in terms of size and dimensions of hidden layers.The developed models investigate the structure of internet traffic time series to grasp the accuracy in predictions. The models are based on stacking of Restricted Boltzmann Machines (RBMs) to create a deep architecture of neural network.To

validate our choice for hidden layer size selection, further more experiments were done for chaotic time series prediction.

In this contemporary era, the world of internet is facing an increasing challenge to design more reliable and robust strategies for network communication. The rapid changes in the network topologies bring abundant changes in data traffic. Internet Service Provider (ISP) is an organization that provides services for accessing the internet. Figure 3.1 demonstrates the infrastructure of an ISP. Quality of Service (QoS) is the concept in which error rates, transmission rates and other physical characteristics of the network can be measured, improved, and, to some extent, guaranteed in advance. Analysis of the network traffic load plays a crucial role in optimization of resources, design, management and control of network services [97]. Therefore, internet traffic prediction is very important for the progress of more proficient traffic engineering and anomaly detection tools. On the other hand, the models developed for future traffic forecasting do support the detection of anomalies residing in the data networks. Moreover, irregular amount of spam elements and other security attacks can be identified by comparing the actual traffic with the predictions of forecasting models.

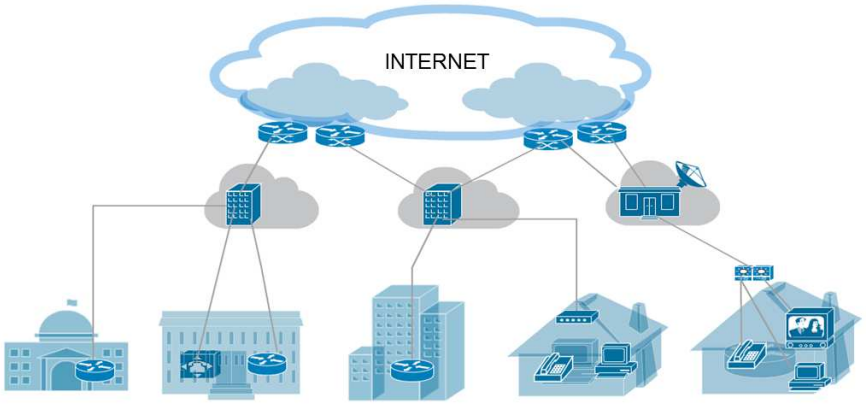

Fig. 3.1 Infastructure of an ISP

Traffic modelling is fundamental to the network performance evaluation. Traffic measurement studies have demonstrated the nature of network load as nonlinear and self-similar [98, 99]. The transportation data is demonstrated as time series with nonlinear and chaotic characteristics [100, 101]. The traffic data in time series is found to be correlated over both short and long time scales. Hence, most of the important correlations and bursting properties are often ignored when traffic is

characterized using simple models. Consequently, our model follows the approach of deep Artificial Neural Network (ANN).

3.2.1 Related Work

It is known fact that the literature review reveals the presence of various models available for time series analysis, prediction and forecasting. Specifically, in this section, we focus on the internet traffic predictions. Internet traffic has been empirically studied by conducting statistical analysis as stated in [102]. The traditional statistical approaches include Box-Jenkins, Autoregressive Moving Average (ARMA) and its variational models. In another research work, the traffic load is also modelled using filtering of non-stationary components and applying the nonlinear threshold autoregressive model [103]. These models are based on discrete linear stochastic processes. Therefore, these models only capture the linear correlation structure present in the time series irrespective of identifying any nonlinear patterns.

Most statistical methods are parametric models that need a great deal of statistical information, whereas ANNs are non-parametric, data-driven and self-adaptive models. The experts in [104] applied the nonlinear time series prediction models for internet traffic prediction and evaluated their effectiveness. These models included Radial Basis Function (RBF) and Support Vector Machine (SVM). A comparative study of Time Series Forecasting methods was conducted on the basis of TCP/IP traffic [105]. Their proposed neural assembled technique as an online forecasting system produced the lowest error as compared to other models. Recent studies demonstrate the implementation of deep ANN models for time series prediction and forecasting [106, 46, 107]. The deep learning model is implemented for internet traffic prediction problem [108]. The stacked autoencoder approach is followed as deep architecture in above mentioned research problems. In [109] a DBN model is developed for the road transportation problem.

3.2.2 Dataset Description

The internet traffic dataset is taken from the Time Series Data Library. The time series dataset consists of a total of 1657 samples recorded hourly from an ISP. The data show aggregated traffic in the United Kingdom academic network backbone

from 2004 to 2005. The dataset contains two attributes, one is traffic load values recorded in bits and the other one is a particular time interval of each record. In order to increase the accuracy of the predictions the input set is further segmented as an hour, month, day, internet traffic at time t, t-1 and t-2. The number of lag terms included as input parameters are calculated by the autocorrelation function. Each input variable is normalized in the range of (0,1). Among the normalized data, the first 1200 samples are chosen to train the model and the rest of the data is used as a test set.

3.2.3 Experimental Setup

The purpose of our work is to address an approach for time series prediction by using the essential attributes of deep learning and its significant strategies for architectural topologies. The proposed models in our approach investigate the structure of internet traffic time series to grasp the accuracy in predictions. The models are based on stacking of Restricted Boltzmann Machines (RBMs) to create a deep architecture of neural network. The layers of an RBM are first trained in an unsupervised fashion. Each layer extracts the nonlinear representation of data. The abstractions of the top hidden layer form highly nonlinear hierarchal features as input set for predicting the network traffic load. In this manner, the estimated future traces of traffic load are predicted at the output layer which is trained in a supervised manner at the fine tuning step of the whole model.

In this section of thesis, we present three different topological architectures of the deep learning model. The proposed deep neural models perform the time series prediction of internet traffic data. The general construction of the input/output variables and structure for hidden layers of deep ANN for the prediction task in each case is subsequently discussed in detail.

One cannot choose the topology of the neural network, i.e. the depth of the network model and the hidden layer size on an arbitrary basis. The researchers in [19] have given emphasis to three different topologies for model development while considering the width selection of hidden layers. This suggests that the preferable choice of model topology is keeping the width of the hidden layers either constant throughout the model or in increasing/decreasing order of size. Therefore, we have optimized the performance of these three different strategies, in order to

select, implement and train the best architecture of deep hierarchical model.The preferred parameter settings for the each model are shown in Table 3.1. Therefore, the architecture of our model is based on decreasing the hidden layer size from bottom to top. The number of neurons in the first hidden layer is calculated through Monte Carlo simulation from the range (100-600).

Table 3.1 Optimal Parameter settings for pretraining.

S.No.	Parameters	Value
1	Transfer function	Sigmoid
2	Batch size	5
3	Learning rate	0.1
4	Initial momentum	0.5
5	Final momentum	0.9
6	Max. No. Iterations	100

Model I

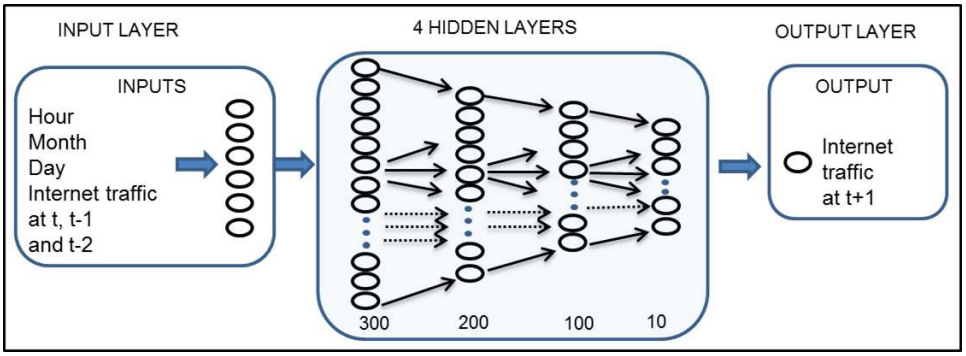

Fig. 3.2 Deep Beleif Network for Model I

Model II

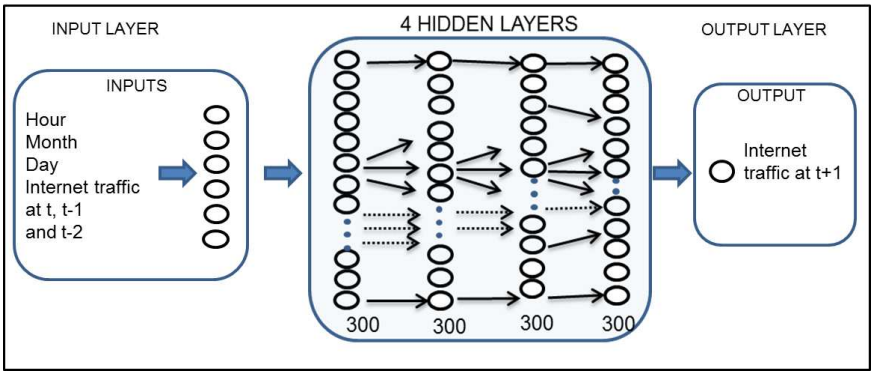

Fig. 3.3 Deep Beleif Network for Model II

Model III

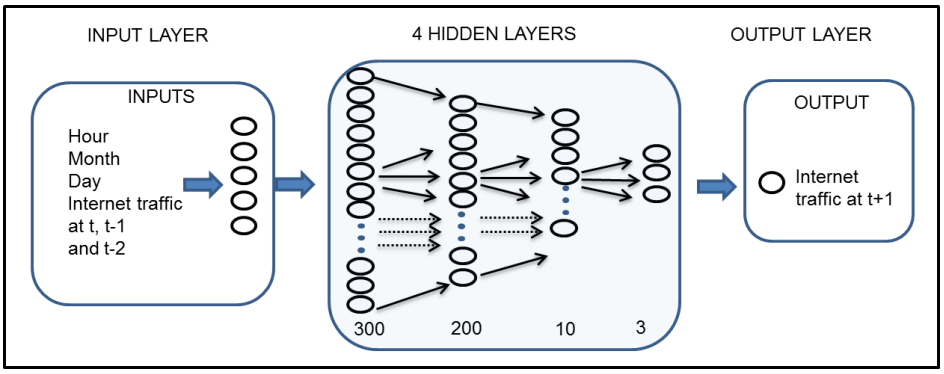

Fig. 3.4 Deep Beleif Network for Model III

3.3 Performance Validation

In order to validate our strategy for hidden layer width selection of deeper neural networks, we conducted two more experiments on Lorenz and Mackey-Glass chaotic time series benchmarks. Two different models, one for Lorenz and other for Mackey-Glass were developed and trained on both series respectively in order to predict the pattern of these simulated chaotic time series. We chose these two particular nonlinear, dynamical and deterministic systems to generate the chaotic behaviour.

3.3.1 Lorenz behaviour

The lorenz system is composed of differential equations which are useful for producing chaotic solutions for some specific parameters and initial conditions. The simplified mathematical model as first studied by Edward lorenz, which consist of three ordinary differential equations, as presented below in (3.1), (3.2) and (3.3).

$$\frac{dx}{dt} = \sigma(y - x) \tag{3.1}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{3.2}$$

$$\frac{dy}{dt} = xy - \beta z \tag{3.3}$$

where, x, y, and z assemble the system state, t is time, σ , ρ , β are the system parameters. However, for our study the equation (3.1) was taken in to consideration. as shown in figure, The lorenz equations are capable to represent the dynamics of lasers, electric circuits, brushless DC motors, chemical reactions and forward osmosis.

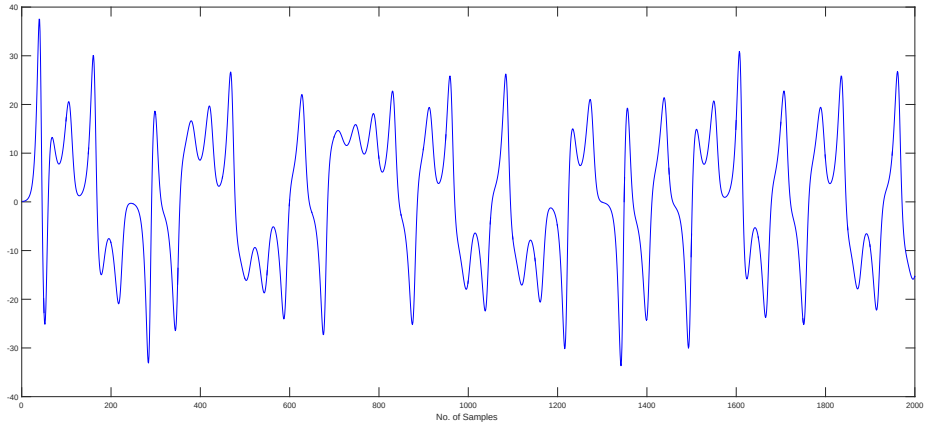

Fig. 3.5 Time series of synthesized Lorenz signal

3.3.2 Mackey-Glass behaviour

The Mackey-Glass time series is the nonlinear time delay differential equation (3.4). Here, τ is the time delay whereas, α and β are system parameters. In our study, the time series signal was generated by using the discretized version as expressed in equation (3.5)

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1+x(t-\tau)^{10}} \tag{3.4}$$

$$x(i+1) = x(i) + ax(i-s)/(1+x(i-s)^{c}) - bx(i) \tag{3.5}$$

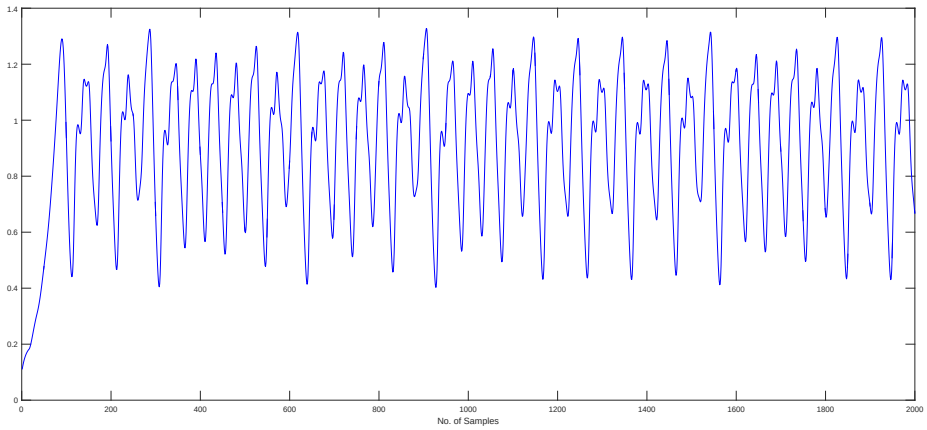

Fig. 3.6 Time series of synthesized Mackey-Glass signal

3.3.3 Validation

In order to verify and validate our strategy for hidden layer selection, we trained two more models one for Lorenzo time series and another for Mackey-Glass as mentioned earlier. The hidden width architecture of both the models is similar as (600-200-100-10) moving from lower to higher level layers . The four lag terms as input variables were given to the input layer and 1 step ahead output was predicted by each model. The training and test results are shown in Table 3.2. These outcomes estimated from both models demonstrate that, our followed strategy for the topology of deep networks specifically DBN's is capable to produce outstanding predictions.

Table 3.2 Performance measures for predicting Lorenz and Mackey-Glass time series

Deep ANN (DBN)	RMSE	
	Training	Test
Model Lorenzo	0.0016	0.0015
Model Mackey-Glass	0.0028	0.0023

3.3.4 Results, Analysis and Discussion

To identify the best topological structure of deep ANN for internet traffic prediction, the three different models of DBN were developed as discussed earlier. The performance measure of each deep ANN model is given in Table 3.3 in terms of Mean Square Error (MSE) and Root Mean Square Error (RMSE). All of the models achieved appreciable performance, but comparatively the first model produces more accurate predictions. This indicates that even for deeper architectures of ANN the proper selection of the number of hidden neurons on each layer is important, otherwise searching for the proper parameter space to initialize the network model may become difficult.

Table 3.3 Performance measures of Model I, II and III

Deep ANN (DBN)	RMSE		MSE		R
	Training	Test	Training	Test	Test
Model I	0.0300	0.0286	8.9e-04	8.1e-04	0.987
Model II	0.0319	0.0310	0.0010	9.5e-04	0.984
Model II	0.0331	0.0427	0.0011	0.0018	0.976

As it is seen in the Table 3.3, the values of performance measures specify the high quality results obtained from model III. In this model the test error is larger than the training error. This is because of the small size of the top hidden layers. There should be enough neurons in the top hidden layers to bring the training error closer to 0 as implemented in the model I. Consequently, due to unsupervised pretraining the test errors are lower than the training errors in the model I and model II, this observation indicate consistency with previous research [18,19]. It is clearly visible from the measures of table 2 values that, the estimated error values of model I and II are quite close to each other. This justifies the fact that with the appropriate hidden

neurons in deeper networks, unsupervised pretraining works as a data-dependent regularizer.

It is important to reiterate that the topology of the DBN implemented in the model I was found to perform better than the other two. Henceforth, the outcomes from this model are focused for further explanation. Figure 3.7 presents the structure and nonlinear trend present in the internet traffic data series. Figure 3.8 illustrates the RMSE performance during the unsupervised pretraining of each layer, starting with the input layer.

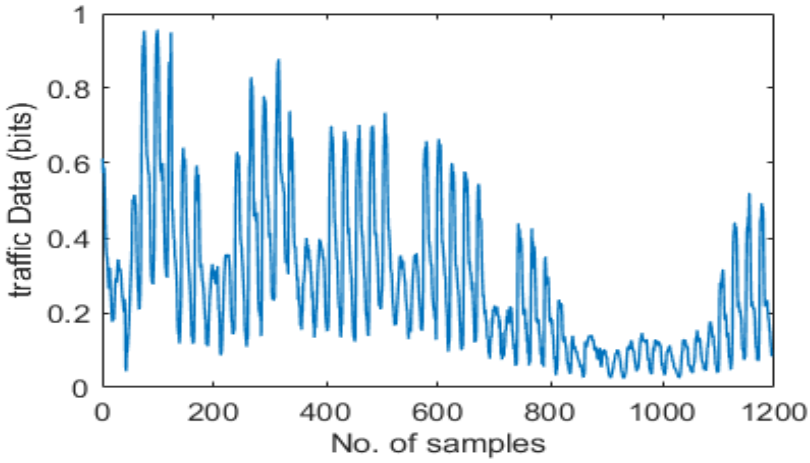

Fig. 3.7 Internet traffic time series

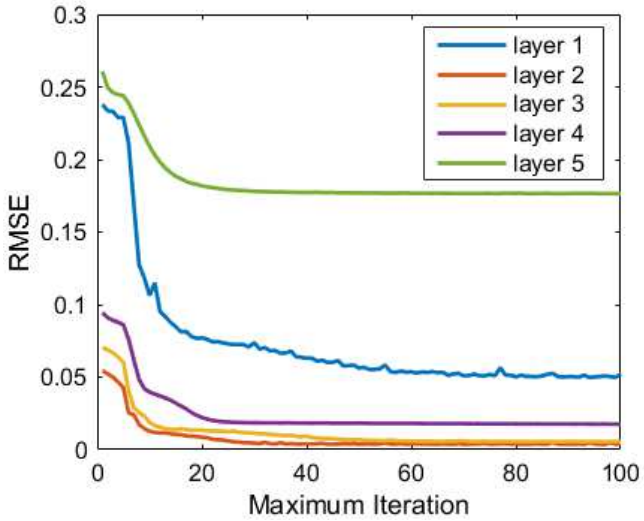

Fig. 3.8 Unsupervised pretraining of hidden layers

In [110], the authors discussed three techniques for visualizing deep layers: activation maximization, sampling from an arbitrary unit and linear combination technique. We have used sampling based method which aims to provide the distribution similar to training set. This may undergo for further processing to make sense of what is captured at chosen hidden unit. The feature learning at multiple levels of abstractions allows a neural network to absorb complex function mappings of input distributions, with the exception of any hand-engineered features. Figure 3.9 presents the learnt features on each hidden layer of the model. It is evident that these learnt features are distributed representations of raw input.

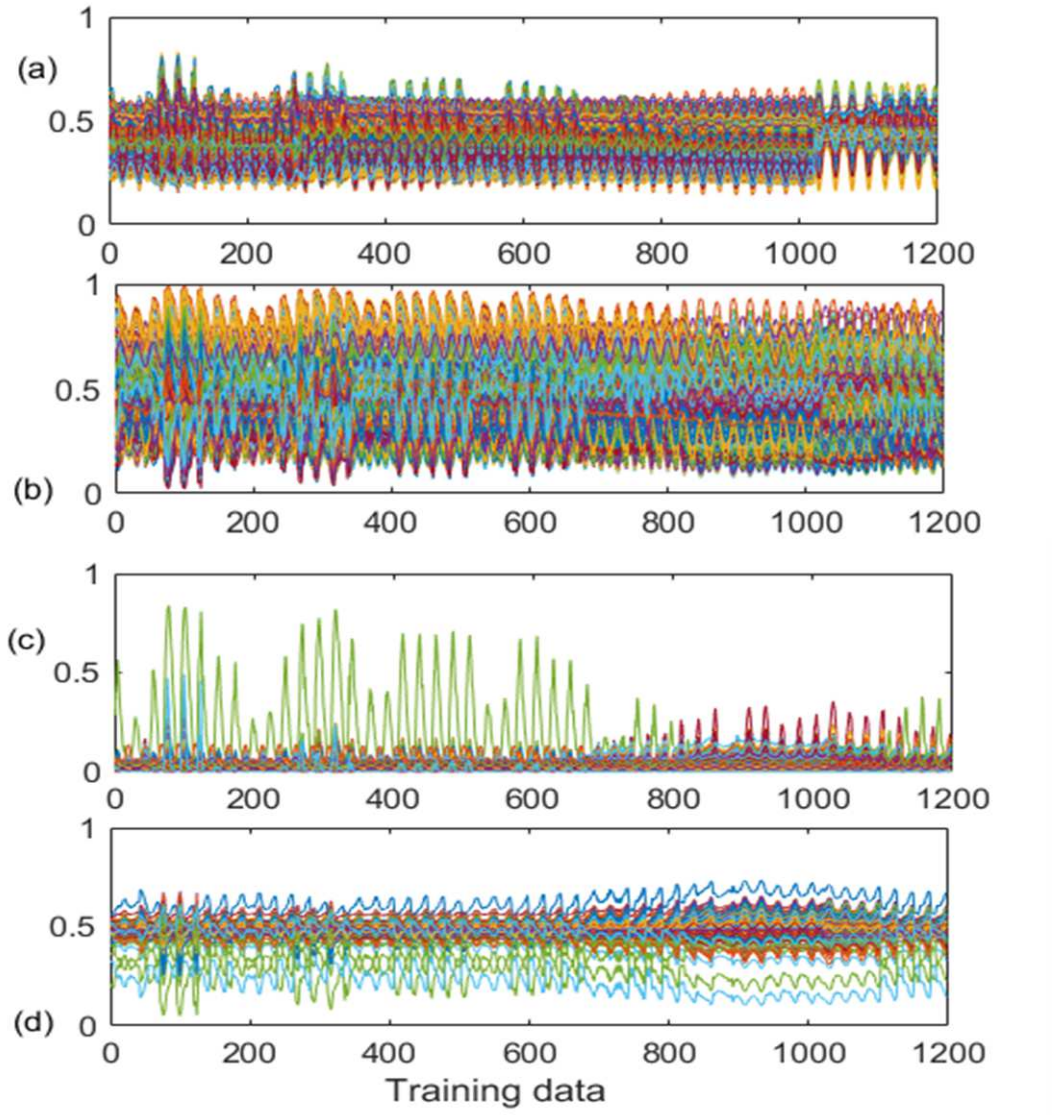

Fig. 3.9 Nonlinear representations as learnt features of hidden layers from Model I

Figure 3.10 presents the distribution of errors estimated by model I on the training data set in form of histogram. It can be seen that the histogram is symmetric, and the errors are mostly centered near 0. The predictions of traffic load estimated by model I, along with the original traffic values are drawn in Figure 3.11 . The model resulted as RMSE of 0.0286 on test data for 457 samples. Thus the equivalent MAPE efficiency is 7.4% on test set. The regression value of R obtained on test data by model I is 0.98, as given in Table 3.3.

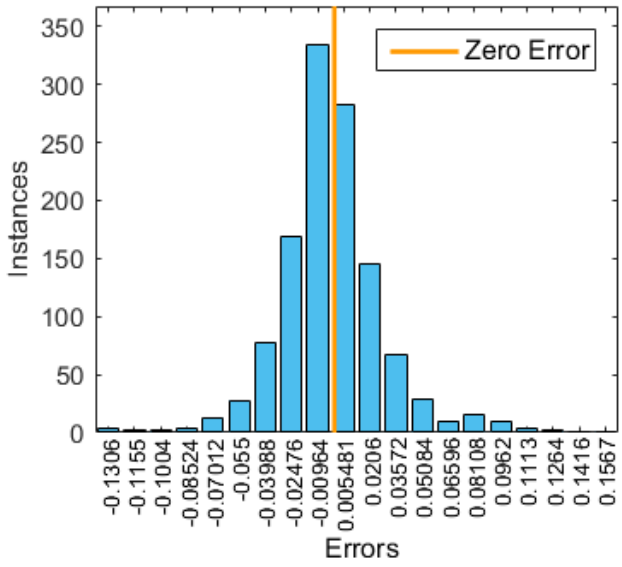

Fig. 3.10 Histogram of errors on trained data

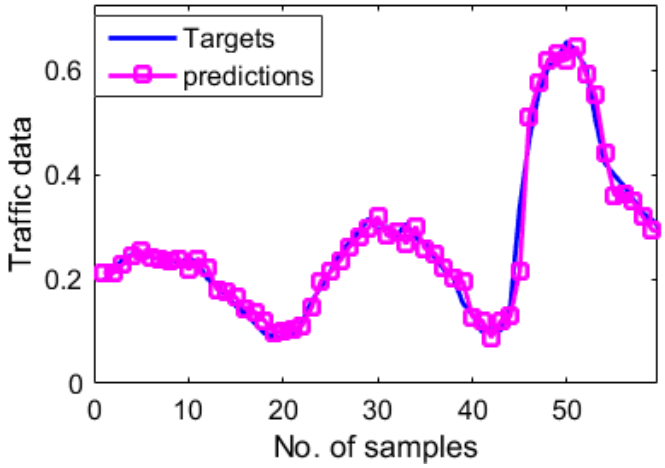

Fig. 3.11 Sixty samples of prediction estimated by Model I (from test data set)

3.4 Meteorological Nowcasting

The activity conducted in this research work is somehow related with our previous work done in [111]. In above mentioned research, a statistical neural system was used to "nowcast" meteorological data measured by a weather station deployed at Neuronica laboratory, Politecnico Di Torino, see Figure 3.12. For further details please refer [111], [112]. By utilizing the same resources of meteorological data, i.e, "NEMEFO", now we have done weather parameter prediction by using deep learning algorithm.

Fig. 3.12 METEO Weather station.

In the first part of this chapter, we presented predictive models for internet traffic prediction by using DBN. We explored the useful strategies for topological architecture for deeper networks and we also did validation on standard benchmark time series. keeping all those aspects in mind, which we earned for the successful training of deep models, we were motivated to attempt some more case studies for real time data sets. Weather forecasting has been one of the most challenging problem around the world for more than a half century. It makes difficult for traditional mathematical or statistical models to adapt irregular patterns of data which can not be written in form of function, or deduced from a formula. In response to this, we trained some more DBN models for air temperature, relative humidity, air pressure and rainfall prediction for the next future value. The pictorial view of our contributed activity is presented in Figure 3.13 which we presented in DET poster day at Politecnico Di Torino.

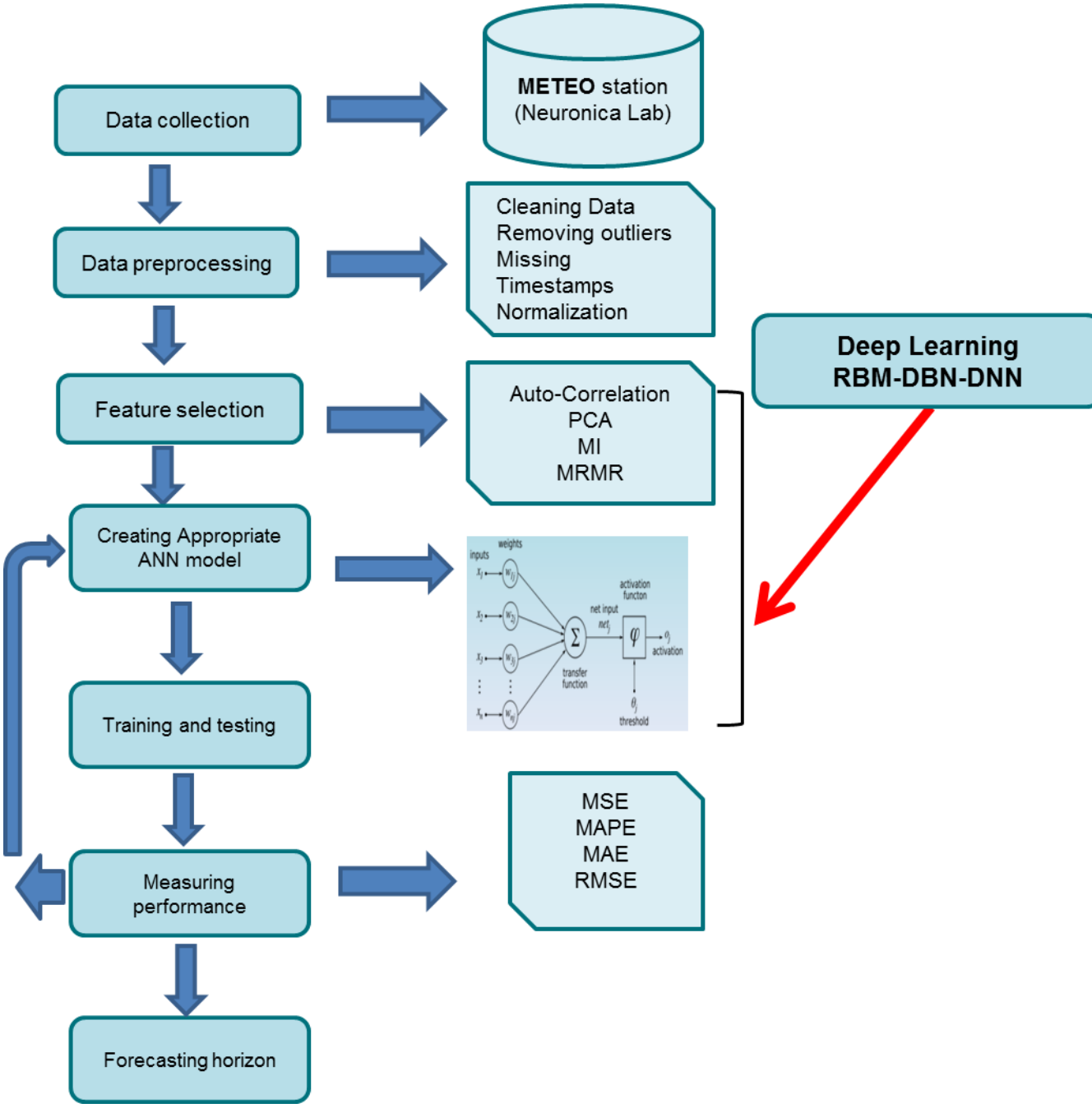

Fig. 3.13 Illustration of Contributed Activity for Weather Nowcasting

3.4.1 METEO Weather Station and NEMEFO

Weather forecasting is a complicated and one of the most challenging task that includes observing and processing huge amount of data. NEMEFO stands for NEural MEteorological FOrecasts. It is basically a software tool connected to Meteo weather station at Neuronica laboratory, which samples Meteo data after every 15 minutes. Meteo station contains following recorded weather data.

- Air temperature
- Relative Humidity
- Air Pressure
- Solar radiation
- Wind velocity
- Precipitation
- Wind Direction
- Corresponding Date and Time

The sensors at Meteo station provide a new recording after every fifteen minutes. The dataset was downloaded which contains the records from 4 October 2010 to 3 September 2015. However, the predictive models are only implemented for nowcasting of Air temperature, Relative humidity, Air pressure and Rain fall as mentioned previously. The data recorded through sensors may have noise, some of missing samples and unwanted frequency fluctuations. In order to detect the outliers and to remove sensor noise, some of the pre-processing in the form of signal filtering has been done on the data prior to considering it as an input set. Subsequently, features are extracted individually for each case to be predicted for next one hour. The dataset was downloaded which contains the records from 4th October 2010 to 3rd September 2015.

3.4.2 Air Temperature prediction

Temperature is one of the most common parameters for an accurate weather forecast. The unit of recorded temperature at Meteo station is Celsius. It is one of the known

fact that the temperature gets effected by season. For example, in extreme summer we face scorching heat by sun and in winter we experience freezing cold temperature. Consequently, the recorded temperature has maximum and minimum values. Apart from this the second effecting parameter could be the particular hours in a day; At that time the air temperature can possibly vary, i.e, the day time hours and the night time hours. Since temperature is clearly dependent on the season and hour, these two attributes have been taken into account in order to reach a right nowcasting. Month and Hour have been computed using the date of the record and they have been used as predictors. They have been preprocessed in order to transform them as sinusoidal features as shown in figure 3.14 and figure 3.15 respectively.

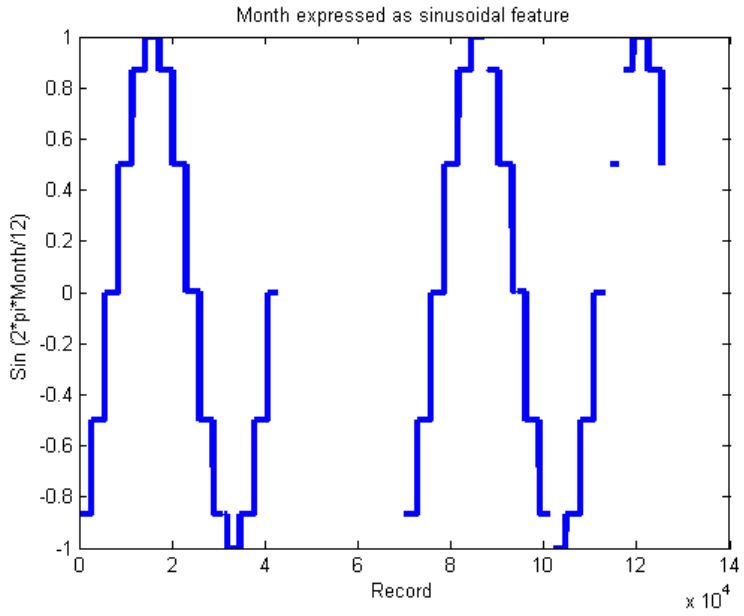

Fig. 3.14 Recorded months converted into Sine waveform

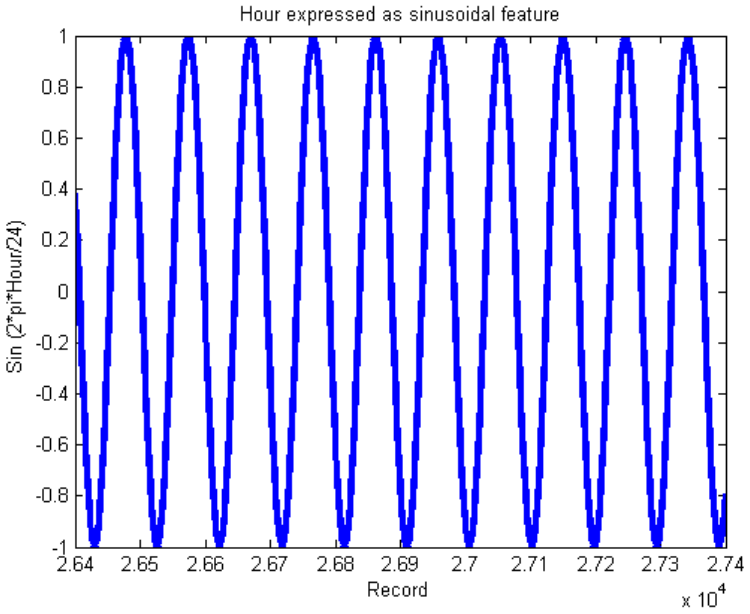

Fig. 3.15 Recorded hours converted into Sine waveform

In order to predict temperature at time t+1, the final input feature set contains particular values of month, hour, temperature at interval (t) and temperature at (t-1). Though before taking the temperature as attribute, we have done preprocessing to reduce the noisy fluctuations from raw sensor data which includes Butterworth lowpass filter with order 2 and 0.11 cutoff frequency in mHz. The difference between actual and filtered data can be seen in Figure 3.16. Moreover, the identified outliers in series were replaced by NAN. Additionally, the interpolation method was applied to cover the missing samples where sensor was unable to record the samples.

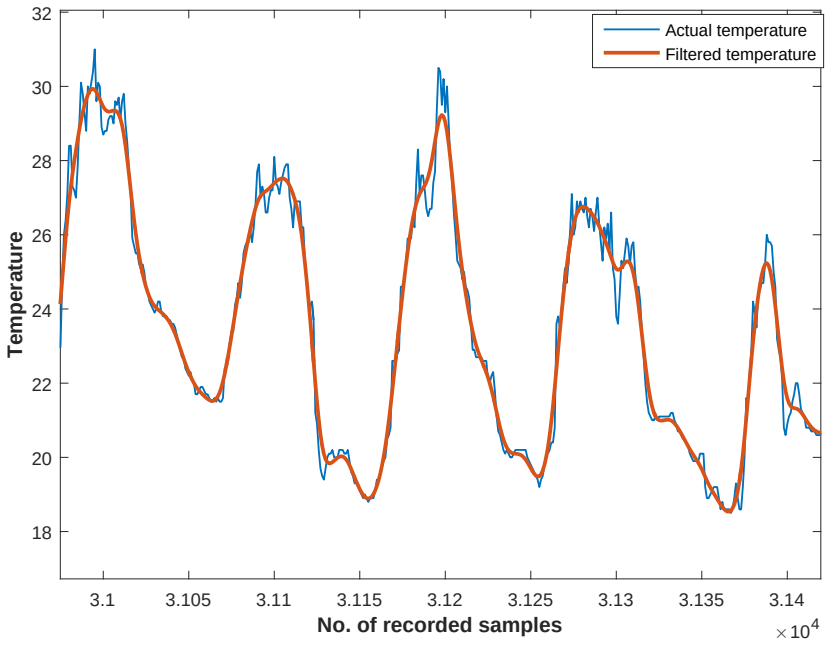

Fig. 3.16 Actual and filtered temperature Series

The most noticeable step is that the input data as well as labels were normalized in the range of (0,1). Because, we are using RBM which deals with binary hidden and visible units. The detail explanation related to this, has already been demonstrated in section 2.2.2. The training data was selected from October 2010 to March 2014. As aforementioned, the input data set consist of five attributes. According to this, the input layer was based on five nodes, whereas, the output layer with one output neuron. For the selection of number of hidden layers and the size of hidden units in each layer, we preferred random search method. In response to this, we developed and trained several architectures. The selection of hyper-parameter for this model and the next upcoming models 3.4.4 and 3.4.3 was based on our earlier hypothesis which provided great support to select better model. The best predictive model for temperature prediction which was initially pretrained layer by layer with total four hidden layers with dimension (500-200-100-10). After training each layer separately the model was trained globally by adding an output layer with temperature labels. The architecture of model is illustrated in Figure 3.17. The results are further discussed in section 3.4.5.

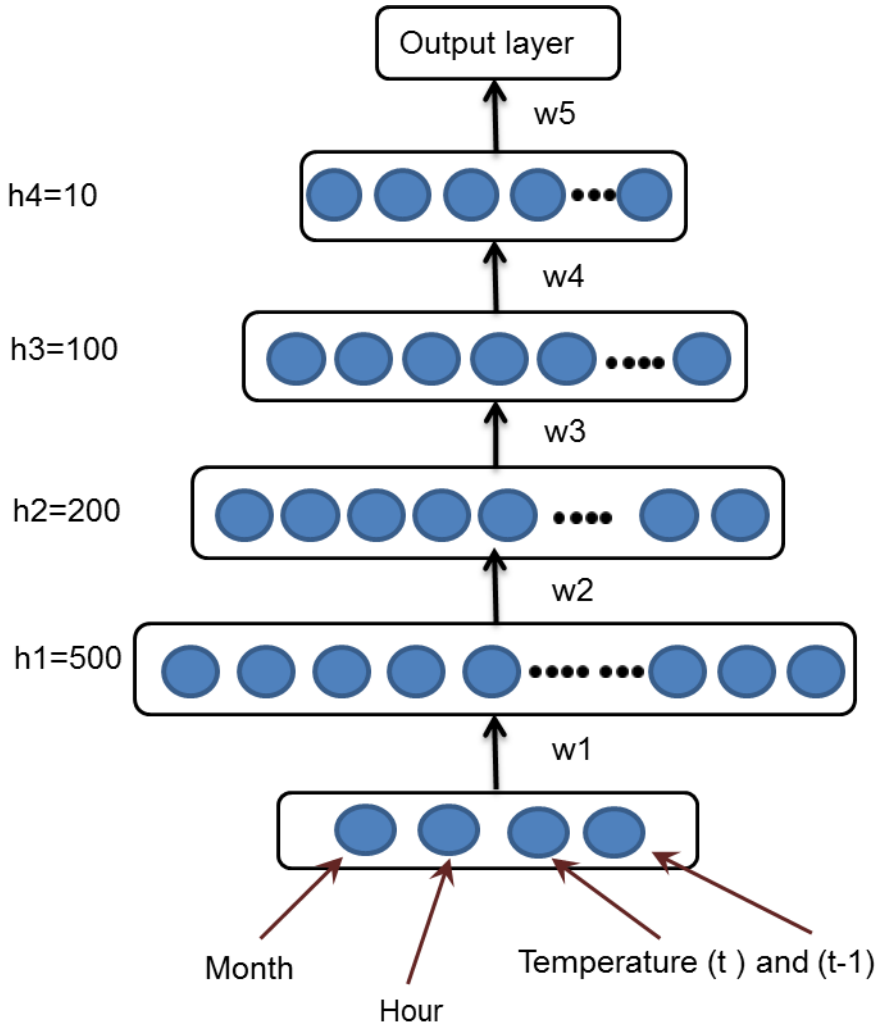

Fig. 3.17 DBN-RBM for Temperature prediction

3.4.3 Relative Humidity prediction

Humidity is a quantity representing the amount of water vapour in the atmosphere. However, relative humidity depends on temperature and the pressure of the system of interest. The variation of temperature, which has a larger variability, depends on the hour and season. Apart from this, we applied Mutual Information Criteria (MIC) to find the best correlations in between of weather parameters, which further confirmed the above mentioned attributes selection. The correlations between features computed via MIC is presented in Table 3.4.

For selecting the features with the highest relevance to the target class c. Relevance is usually characterized in terms of correlation or mutual information, of which the latter is one of the widely used measures to define dependency of variables

Table 3.4 Feature selection using Mutual Information

Attributes	**Time**	**Temperature**	**Pressure**	**Rain**	**Humidity**	**Wind direction**	**Wind Velocity**
Time	2.0748	0.3748	0.1261	0.0081	0.0246	0.01861	0.0281
Temperature	0.3748	1.933	0.0092	0.0189	0.0189	0.0421	0.0671
Pressure	0.1261	0.0920	1.9787	0.0223	0.0661	0.0227	0.0343
Rain	0.0081	0.0189	0.0223	0.308	0.05	0.01	0.005
Humidity	0.0246	0.0189	0.7256	0.0534	1.8134	0.0415	0.0920
Wind direction	0.0186	0.0421	0.0534	0.0154	0.0415	1.6601	0.0252
Wind velocity	0.0281	0.0671	0.0343	0.0058	0.0920	0.0252	1.7766

Given two random variables x and y, their mutual information is defined in terms of their probabilistic density functions $p(x), p(y)$, and $p(x,y)$:

$$I(x,y) = \int\int p(x,y) log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{3.6}$$

The selected features xi are required, individually, to have the largest mutual information $I(x_i;c)$ with the target class c, reflecting the largest dependency on the target class. In terms of sequential search, the m best individual features, i.e., the top m features in the descent ordering of $I(x_i;c)$, are often selected as the m features [113].

Hence, features used as inputs for the training are corresponding temperature, previous pressure, previous humidity, corresponding Month and Hour. This feature set and labels were further normalized in the range of (0,1) prior to training. The humidity data was filtered with Butterworth filter corresponding same order of 2 and cutoff frequency at 0.11 mHz. The Actual and filtered humidity data is shown in Figure 3.18.

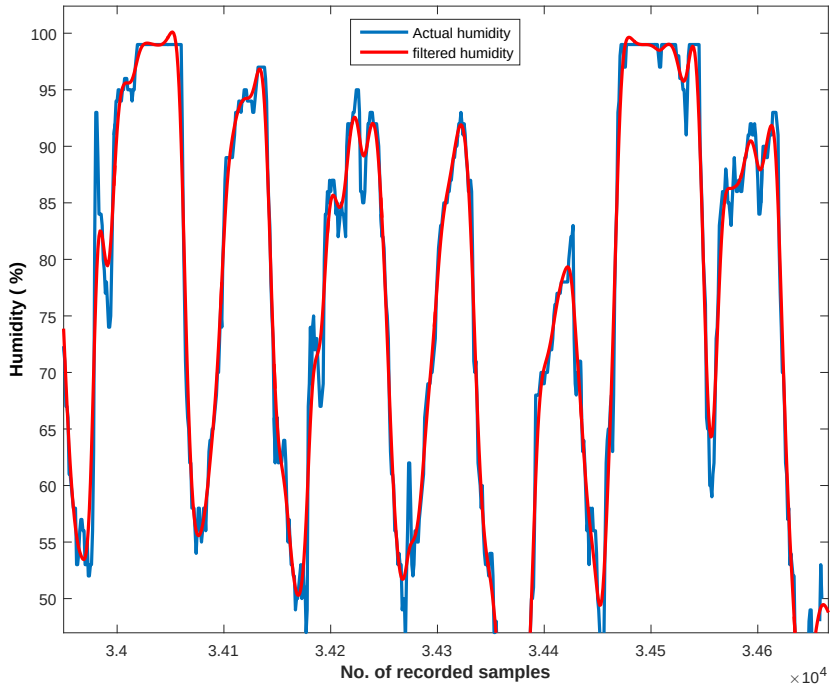

Fig. 3.18 Actual and filtered humidity series

In order to construct and train a predictive model for humidity prediction, a DBN was developed with four hidden layers, one input layer consist of Six nodes and one output layer based on one output neuron. The hidden layers were RBM of size (300-200-100-10), which were trained one by one in a layer wise greedy way with contrastive divergence. The model is illustrated in Figure 3.19. Initially, all weights and biases were assigned the value zero. The model was trained with total 120k samples and was further tested with rest of the data. The pretraining of RBM was performed using minibatches of size 10, with maximum one iteration for each layer pretraining. After training each layer separately the model was trained globally by adding an output layer with normalized humidity labels. However, fine tunning was performed with Maximum 800 iterations. The results are further discussed in section 3.4.5.

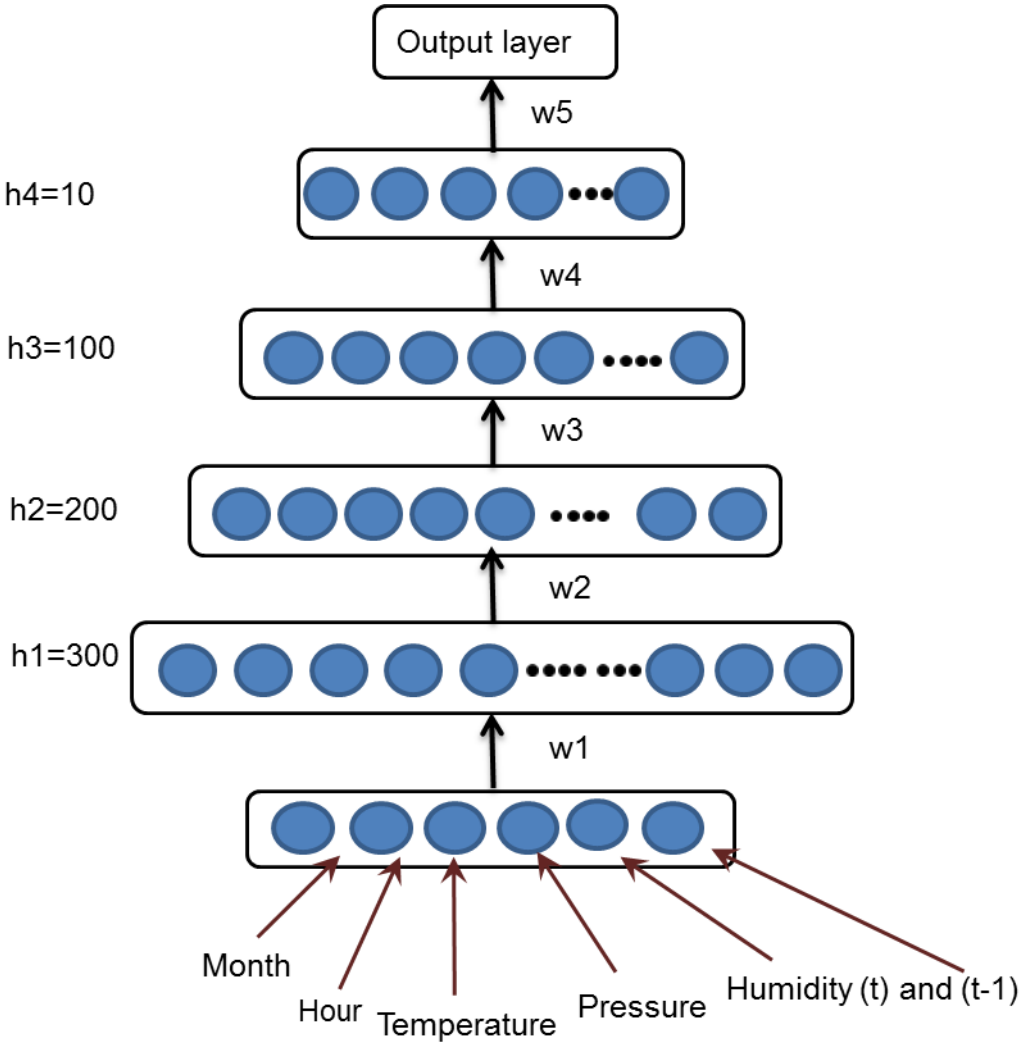

Fig. 3.19 DBN-RBM for Humidity prediction

3.4.4 Pressure prediction

In general, pressure is a force applied perpendicular to the surface of an object per unit area over which that force is distributed. However, atmospheric pressure or air pressure, sometimes also called barometric pressure, is the pressure exerted by the weight of air in the atmosphere of Earth. The pressure data was smoothen with Butterworth low pass filter in same way as air temperature and relative humidity. However, a high pass filter was also applied on the data to detrend the linearly

decreasing trend observed in recorded air pressure series. The figure 3.20 presents the graph of actual and filtered pressure samples.

In order to extract valuable features for pressure prediction we did some in-depth analysis. The main factor that affects the air pressure at a given location is the altitude (or height above sea level) of that location. In order to select the meaningful features for air pressure prediction we did little research on internet. We came to know that the pressure depends on the density or mass of the air. Moreover, the density of air depends on its temperature and from our meteorological dataset the temperature depends on Season (categorized in months) and hour of the day . Thus we took the following parameters as input attributes for predicting the next pressure in series. The month, an hour, corresponding temperature, pressure at (t) and (t-1).

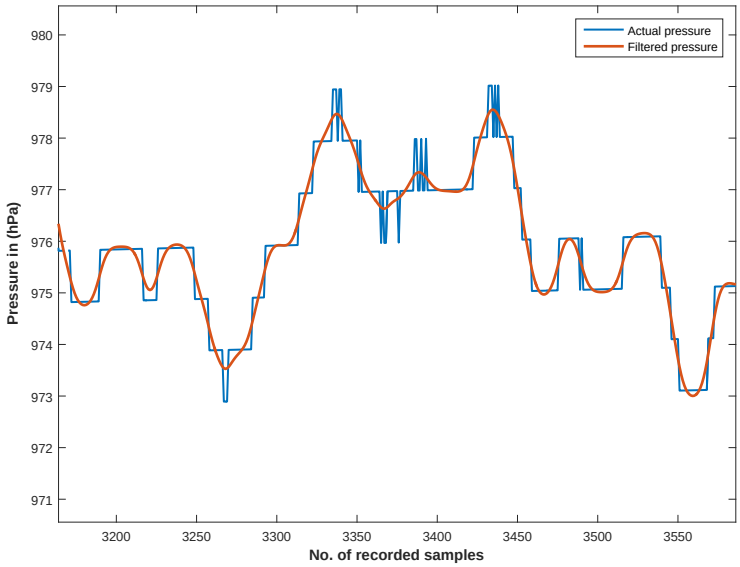

Fig. 3.20 Actual and filtered pressure time series

In order to construct and train a predictive model for pressure prediction, a DBN was developed with three hidden layers, one input layer and one output layer. The hidden layers were RBM of size (300-200-5), which were trained one by one in a layer wise greedy way with contrastive divergence. The model is illustrated in Figure 3.21. The model was trained with total 120k samples and was further tested with rest of the data. The pretraining of RBM was performed using minibatches of size 10, with maximum one iteration for each layer pretraining.After training each layer

separately the model was trained globally by adding an output layer with normalized humidity labels. However, fine tuning was performed with Maximum 800 iterations.

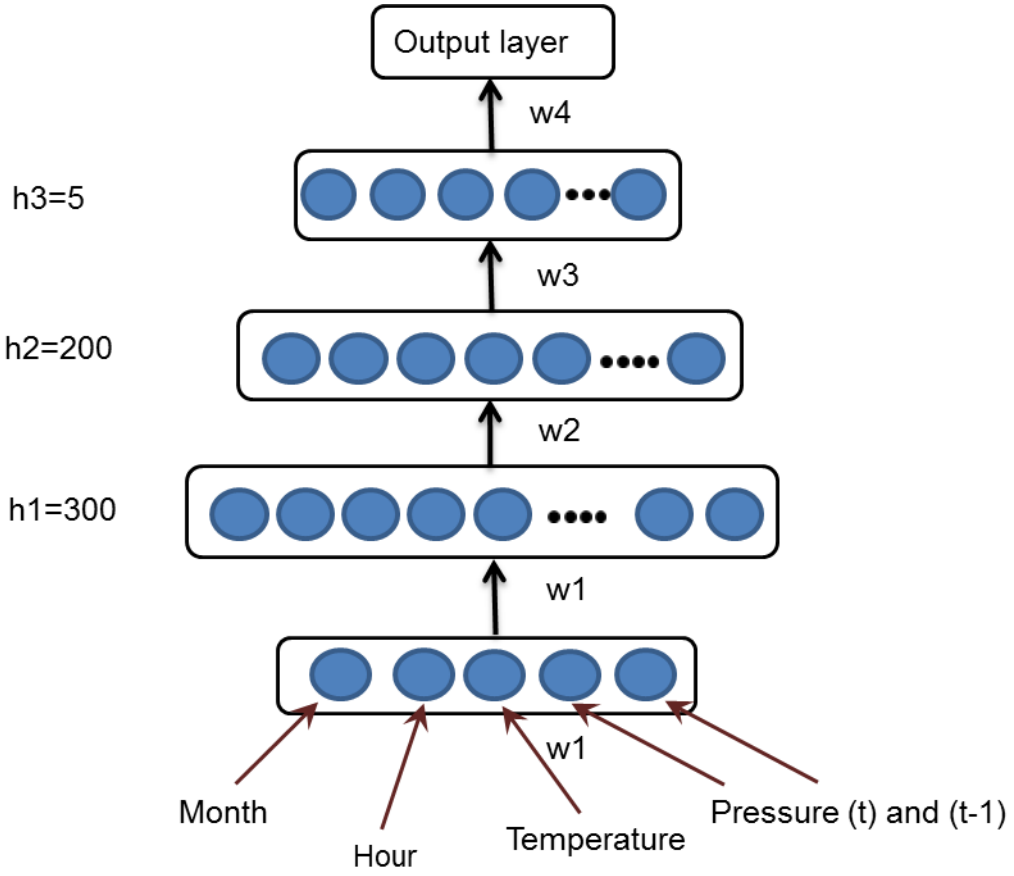

Fig. 3.21 DBN-RBM for Pressure prediction

Initially, all weights and biases were assigned with the value zero in the training of each predictive model case. However, after pretraining the weights are in shape of normally distributed data as shown in Figure 3.22. This identifies that weights are not randomly initialized in order to find the suitable solution. The weights were further transformed during the pretraining phase as shown in Figure 3.23. These weight illustrations are taken from predictive model trained on pressure dataset.

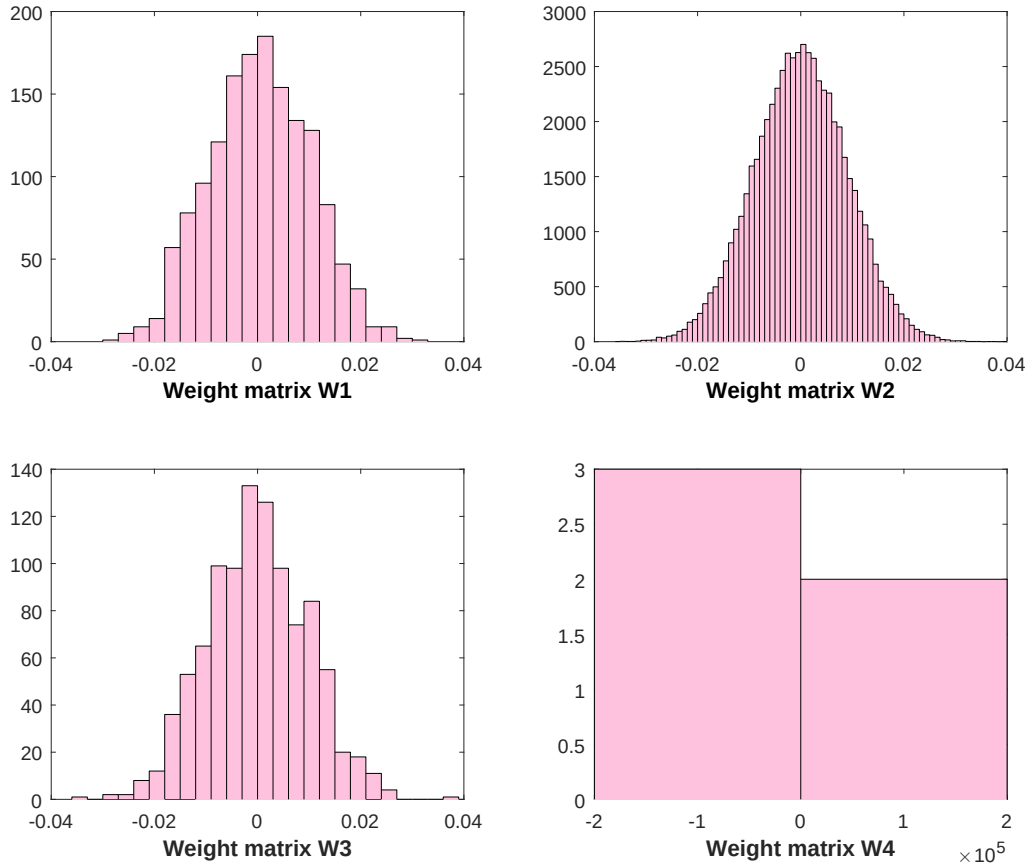

Fig. 3.22 Histograms: showing weights distribution after pretraining step.

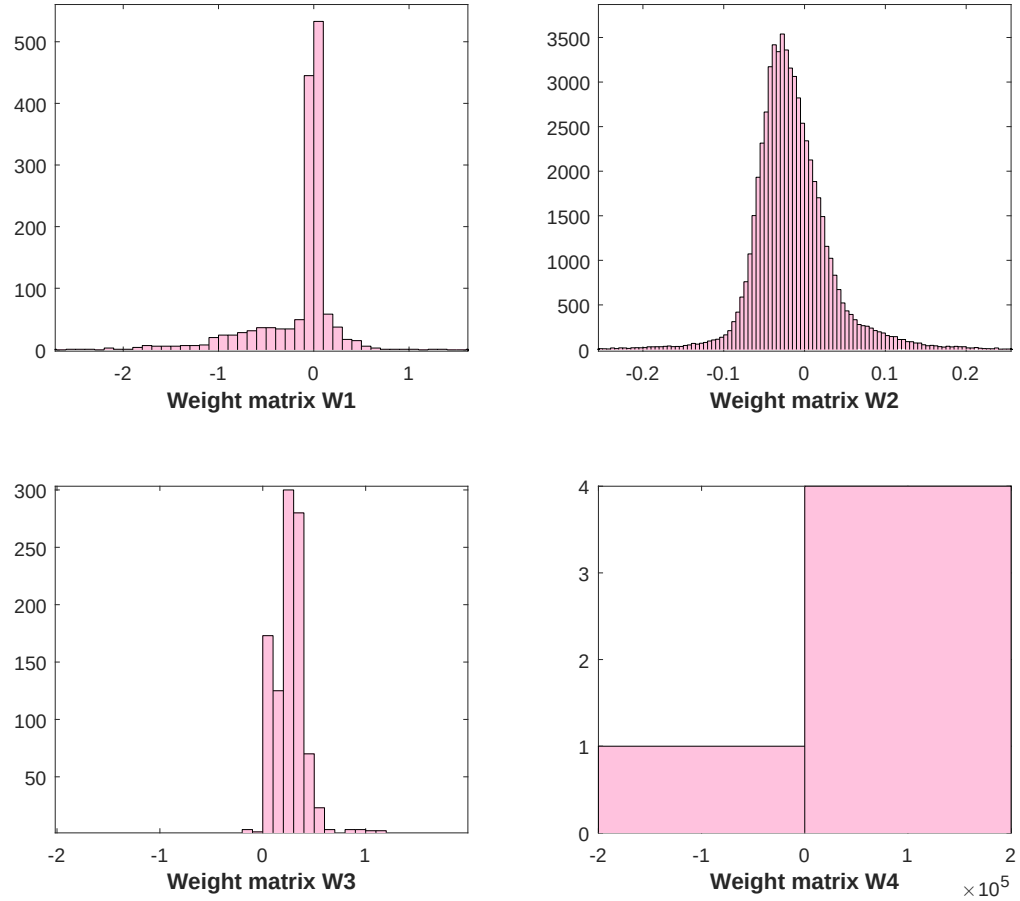

Fig. 3.23 Histograms: showing weights distribution after fine tuning step.

Weights in the hidden layers of a particular model attempts to explore the nonlinear representations or features from the data. In this regard, the computed weights are also termed as feature detectors or receptive fields. These can be considered as a good way of visualizing which kind of features the hidden units have learned. There is a possibility that less meaningful or insignificant detectors may also be present. There is a belief for deep learning provided by practical arguments. In order to capture more abstract and high level structure from data, the top layers or higher layers in deep architectures provide a more statistically salient representation for a particular task. The results are further discussed in next section.

3.4.5 Discussion

In this section, we describe and further discuss in detail the evaluation of trained DBN models for METEO nowcasting. According to our objective in this part of our work we first attempted for next step or next sample prediction of non-stationary time series through DBN. In first part of this chapter as aforementioned, we were successful in deploying and training accurate models for internet traffic prediction even without working much on feature extraction step or its processing and selection. Later on in this chapter, we attempted for METEO nowcasting, weather parameter prediction. In our data set, each weather parameter owns distinct trend and diverse behaviour in its time series. For each parameter, we trained separate model and feature selection was performed accordingly. The performance for each predictive model is presented in following Table 3.5.The performance for each predictive model is measured through MSE , RMSE, R.

Table 3.5 RMSE and MSE on Training and Test sets for Prediction of Metrological Parameters.

DBN Models	RMSE		MSE		R
	Training Error	Test Error	Training Error	Test Error	Test set
Temperature prediction	8.57e-4	9.06e-4	7.35e-7	8.20e-7	0.99
Humidity prediction	1,3e-3	1.5e-3	1.58e-6	2.27e-6	0.99
Pressure prediction	9.07e-4	7.73e-4	8.24e-7	5.98e-7	0.99

In figure 3.24, we present actual and predicted temperature samples taken from test set. It is visible from the figure that predicted samples are highly replicating the original temperature data.

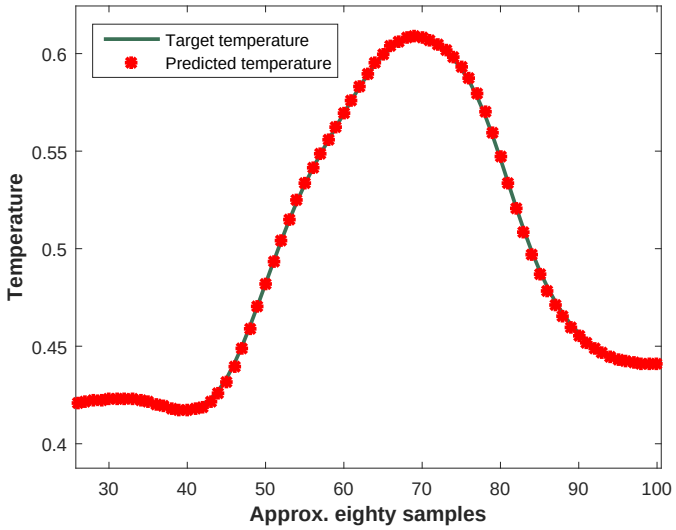

Fig. 3.24 Close view of target and predicted temperature with eighty samples from test dataset.

It is clearly depicted from the Figure 3.25 that, predicted humidity samples of test set are very close to the original recorded humidity. In the same way, strong correlations of predicted and recorded pressure can be seen from Figure 3.26

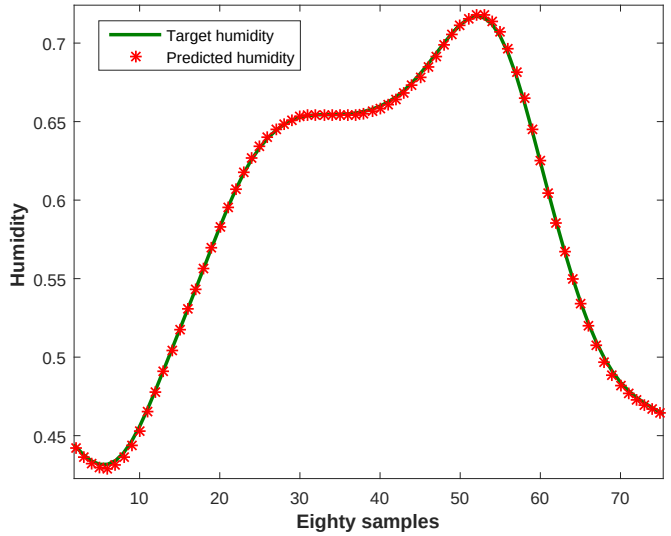

Fig. 3.25 Close view of target and predicted humidity with eighty samples from test dataset.

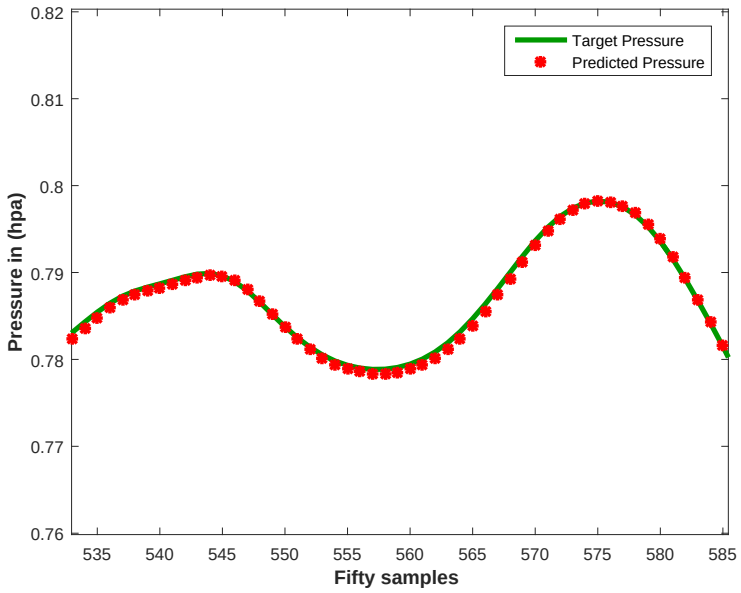

Fig. 3.26 Close view of target and predicted pressure with fifty samples from test dataset.

Comparison with Traditional models

In this section, we demonstrate the comparative evaluation of each aforementioned predictive deep learning model. Several diverse models are taken into consideration. Each of the traditional approach is implemented and trained with different configurations. This was done to select the best tuned model from each of the category. For example, we take NARMAX approach, now various NARMAX models with different number of parameter settings were trained. Out of those trained models , the model with best possible accuracy and performance measure was taken into account to further compare it with our proposed deep learning models. Similarly, we scrutinized NARMAX, NARX, NAR and Recurrent networks for comparative evaluations with deep learning models for time series prediction. We were unable to see some good performance results from the trained RNN architectures. For the choice of RNNs, we trained layer recurrent architectures. Unfortunately, the settings we chose to trained those models were resulting in highly overfitted models. For each meteorological parameter, we trained several layer recurrent models.

Table 3.6 Error Comparison with other traditional approaches for Temperature prediction.

MODELS	ERROR VALUE (RMSE)	
	Training Error	Test Error
DBN-DNN	0.000857	0.000906
NARMAX	0.0013	0.0014
NARX	0.0022	0.0021
NAR	0.0023	0.0022
RNN	0.00	0.00
LSTM	0.11	0.09

Table 3.7 Error Comparison with other traditional approaches for Humidity prediction.

MODELS	ERROR VALUE (RMSE)	
	Training Error	Test Error
DBN-DNN	0.0013	0.0015
NARMAX	0.0052	0.0043
NARX	0.0085	0.0098
NAR	0.0070	0.0079
LSTM	O.24	0.25
RNN	2.08e-5	0.42885

Table 3.8 Error Comparison with other traditional approaches for Pressure prediction.

MODELS	ERROR VALUE (RMSE)	
	Training Error	Test Error
DBN-DNN	0.000907	0.000773
NARMAX	0.0237	0.0046
NARX	0.0236	0.0213
NAR	0.0270	0.0031
LSTM	0.01	0.01
RNN	0.0077	0.023

The Table 3.6 presents the measure statistics of RMSE on all trained models for temperature prediction. It shows that DBN-DNN achieved the best accuracy yielding the least error values on both training and test sets. Similarly for humidity prediction, Table 3.7 presents the error results obtained from the models trained over humidity data. Error measures for comparative evaluations of pressure prediction are summarised in Table 3.8.

The results obtained from models are robust and shows considerably good predictions. Since, the models perform the forecasting task for only next one sample, however for meteo nowcasting our concerned objective is to predict for next three hours. In order to accomplish this task, we further modified our basic approach for time series prediction. This results as multistep prediction through DBN model. In response to this, we considered some temporal statistics and properties , which is further presented in chapter 6. Although, we have done multi-step forecasting in Chapter 4, but the approach is different in both.

3.5 Conclusion

This chapter basically contains two different but progressive parts of our research. In the first part, an application of internet traffic prediction with DBN is implemented. Supplementary to this, current part also provides investigation for the selection criteria of hidden layer width in deep neural networks. In order to identify the suitable topological structure, three different prediction models of deep architecture were developed. Depending on their comparative analysis, it was found that the number of neurons in the hidden layers is crucial for deeper networks. The model with the decreasing width of hidden layers, albeit with sufficient hidden units in higher layers were found to be a proficient predictive model. The results indicates that Model I and II are capable to accurately capture the salient characteristics of the network traffic. To further justify the feasibility of our approach for hidden layer size strategy, experiments were performed for chaotic time series prediction which also produced promising results. For this purpose, the benchmark series were formed by using Lorenz and Mackey-Glass equations. The trained DBN-DNN provided good prediction accuracy. The data used here, were simulated series not real world data. However, the real world data is often complicated with anomalies and change points, which can deviate the trained models from capturing the underlying patterns

of time series. Thus, the subsequent part of this chapter introduces a predictive ANN model based on deep learning hierarchical architecture, for the prediction of real time weather parameters such as temperature, pressure and humidity. The results shows outstanding performance of implemented DBN models while producing the accurate estimations. Furthermore, some comparison evaluations are conducted corresponding with other traditional approaches.

Chapter 4

Hybrid Approach for multi step ahead forecasting

4.1 Outline

In the previous chapter, the implemented forecasting approach was capable to perform one-step ahead predictions. However, the multi step ahead forecasting is much more challenging task then one step ahead. The multi-step forecasting is more frequent and of greater importance in the majority of applications. Therefore, this chapter presents a new method for multistep time series forecasting. This proposed novel approach is implemented for forecasting the future values of time series data. Part of the work described in this chapter has been previously published in [114]. The proposed approach comprises on the combination of two different modelling aspects as a hybrid model. It actually performs the feature extraction of the input time series through deep learning approach and uses a nonlinear autoregressive model for multistep ahead predictions.

One of the foremost focus of deep learning algorithms is to automatically discover significant attributes of data as features. These features are actually exaggerated from hidden layers and are further transformed from as simplest level features to higher level complex representations of inputs [75]. This ability of deep architectures, to automatically learn the powerful features without using any hand engineered human effort or statistical approach is becoming increasingly popular as the range of applications in machine learning discipline continues to propagate. Temperature

is one of the most common parameters for an accurate weather forecast and it has therefore, been selected as a case study for the current work. However, the methodology must be considered as general and applicable to different and larger sets of meteorological parameters.

In order to deal with multi-step time series forecasting problem, one can have choice to opt for direct or recursive forecasting strategy. The use of recursive forecasting strategy introduces the concept of iterations. This means that forecasts are computed by iteratively applying 1-step ahead model for desired number of steps. On the other hand, in direct strategy, the forecasts are estimated directly by implemented model for each forecast horizon. This chapter deals with the multi-step forecasting by following the recursive approach.

In this part of our research work, a novel approach is introduced for forecasting the future values of time series data. The work is based upon the feature extraction of input time series through a deep learning model and nonlinear autoregressive model is implemented and trained on those representations learned by deep network. This later model is developed to execute the task of multistep prediction for future samples. Meaningful features are extracted from the recorded temperature data series by developing and training Deep Architecture NN, specifically DBN (Deep Belief Network).

Smart and intelligent hybrid models that are basically combination of different intelligent algorithms are proposed in the literature with the aim of achieving more robust prediction accuracy [115–118]. This happens due to the amalgamation of the strong characteristics of various methods into a single system. However, due to the difficulty in determining the exact nature of time series, it is often considerably challenging to estimate and compute the appropriate forecasts [119].

4.2 Research Method for Hybrid Approach

In order to compute multistep forecasting, we took the temperature data set of Meteo weather station of Neuronica laboratory. The preprocessing of the recorded data was exactly similar to those steps as mentioned in section 3.4.2 on page No. 45. At first, data was scrutinize for unwanted noise and outliers. Later on, linear interpolation was performed to replace the missing samples followed by an appropriate filtering step.

The significant and noteworthy steps conducted in this methodology are elaborated in Figure 4.1. Additionally, we further test this new approach for multi step forecasting on the simulated chaotic benchmark time series. The proposed approach shows satisfactory results.

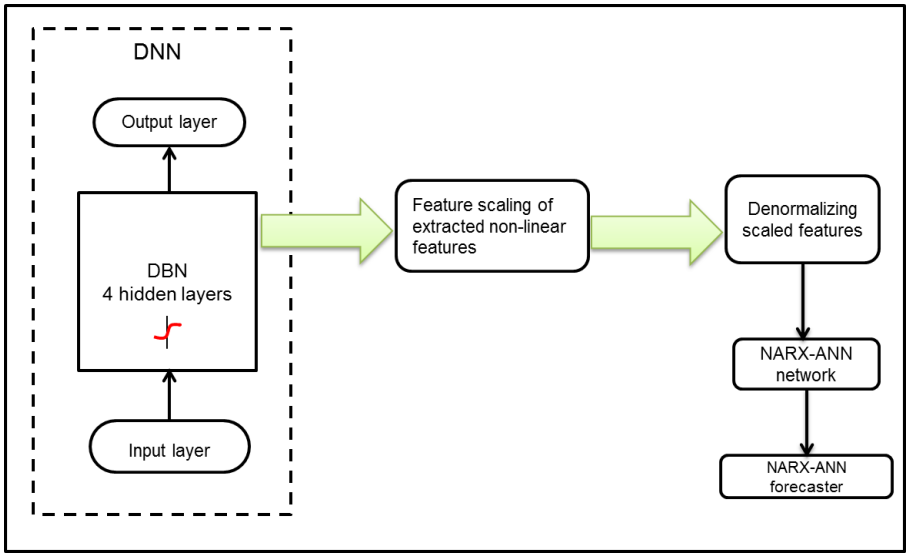

Fig. 4.1 Proposed Method for Multistep Forecasting.

4.3 Experimental Setup

The first part of our work aims to create and develop a DBN model, which is actually able to understand the behaviour of time series and dynamics of structure present in the time series data. The postulated DBN model is trained in a way similar to demonstrated in [77, 76]. This trained DBN model is capable enough to generate the abstract nonlinear features of our recorded time series. The subsequent part of our novel approach deals with the development of a forecasting model. This forecasting model is trained on highly non-linear hierarchical abstractions. The forecasting model is developed to capture the linear dependencies and the nonlinear Auto-regression entity because the time series exist with the natural temporal order of observations. The estimation of future values depends on previous observations available in the record, also including some external effectors and some stochastic term.

4.3.1 Feature Learning with DBN

In the first section of multi step time series forecasting, our objective was to prepare a meaningful set of attributes. In response to this, a neural network architecture based on Six-layers was developed and trained with deep learning algorithm. The DBN model for feature learning contains four hidden layers, as illustrated in 4.2. The least most layer is an input layer which take four attributes of recorded preprocessed data set. The top most layer is the output layer which predicts the temperature data.

The size of each hidden layer was computed on the basis of our analysis and suggestions that we presented in Section 3.1. Each layer in DBN model is treated as RBM. Therefore, these layers were added constructively, one by one. Consequently, the layers were trained independently in an unsupervised way without incorporating the labels initially. The states of hidden nodes computed at each trained RBM were further proceed as input to the next subsequent layer. The specification of parameters used for pretraining of each layer are mentioned in Table 4.1. The pretraining of each layer proceeded with only one epoch. The error and the mean approximations of each corresponding layer are summarized in Table 4.2. Afterwards, the top most layer was added and the labels were placed according to the initial input given at first layer. This recently added last layer was trained in supervised manner.

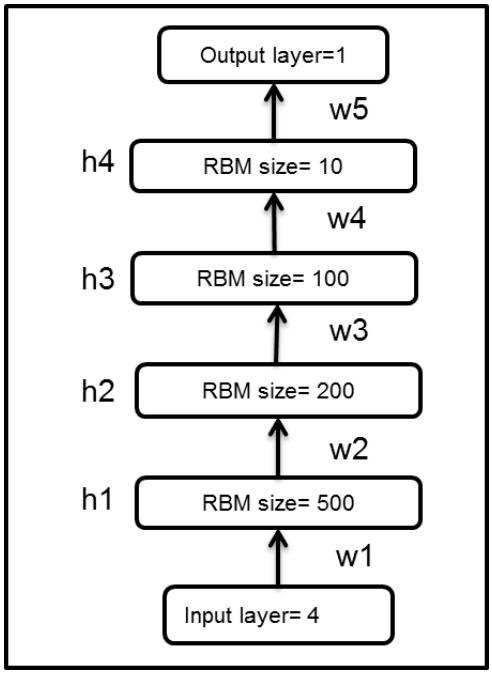

Fig. 4.2 Proposed DBN model for features learning.

Table 4.1 Optimal Parameter settings for pretraining of DBN model.

S.No.	Parameters	Value
1	Activation function	Sigmoid
2	Batch size	5
3	Initial momentum	0.5
4	Final momentum	0.9
5	Learning rate	0.1
6	Max. No. Iterations	1

Table 4.2 Error and Mean Approximation of pretraining.

Layers	RMSE	Mean Value
layer 5	0.0034	5
layer 4	0.0024	0.1
layer 3	0.0011	0.9
layer 2	0.0004	0.5
layer 1	0.2744	1

After completing the pretraining, the reformed outcomes as weight distributions associated with each layer are shown in Figure 4.3. It is clearly perceptible that pretraining initializes the weights and biases of the network to sensible values. The structure of weights connected with each layer as feature detectors of the model is visible in Figure 4.4. This parameter initialization with unsupervised greedy layer-wise training expresses that weights linked with each layer are efficient but although not optimal ones. After unsupervised greedy layer-wise training, fine tuning was applied by backpropagation algorithm, training the model with stochastic gradient descent. The pre-training gave a good start to retrain the model by driving the loss function towards its minima. A total 800 of iterations were used to train the pre initialized DBN model. The weights of model influenced by fine tuning are now modified into different depictions than the earlier ones. The patterns in weight matrices of DBN are illustrated in Figure 4.5.

In order to train the DBN model as DNN, the fine tuning procedure took more than 24 hours. Although, 16 and 32 cores of HPC cluster AMD Buldozer CPU were used for this research.

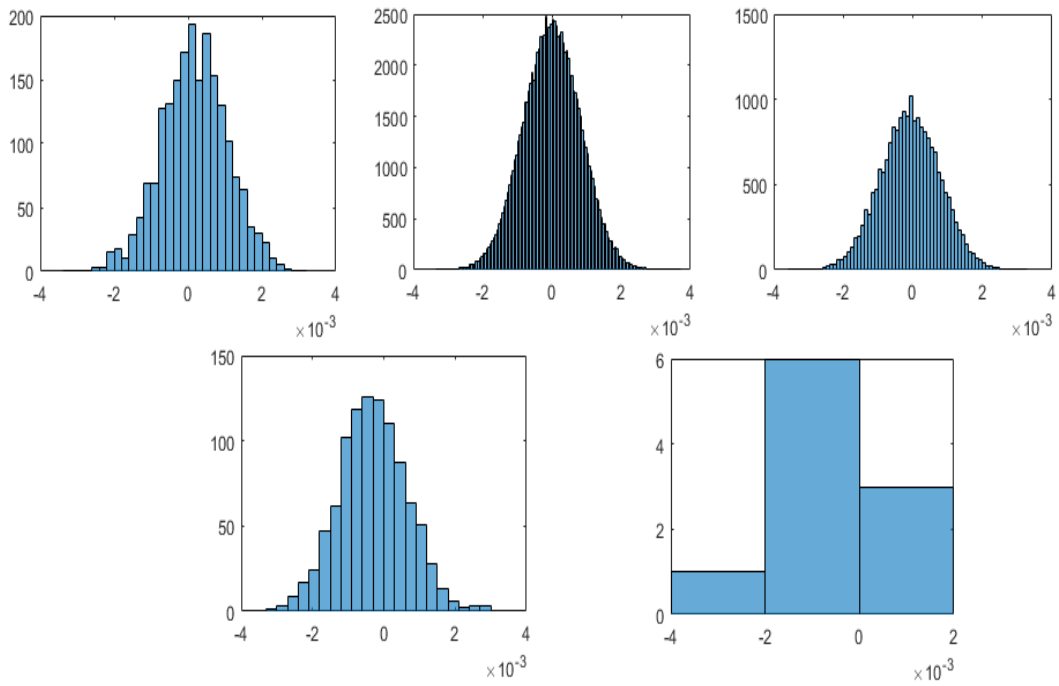

Fig. 4.3 Weights distribution of pretrained DBN

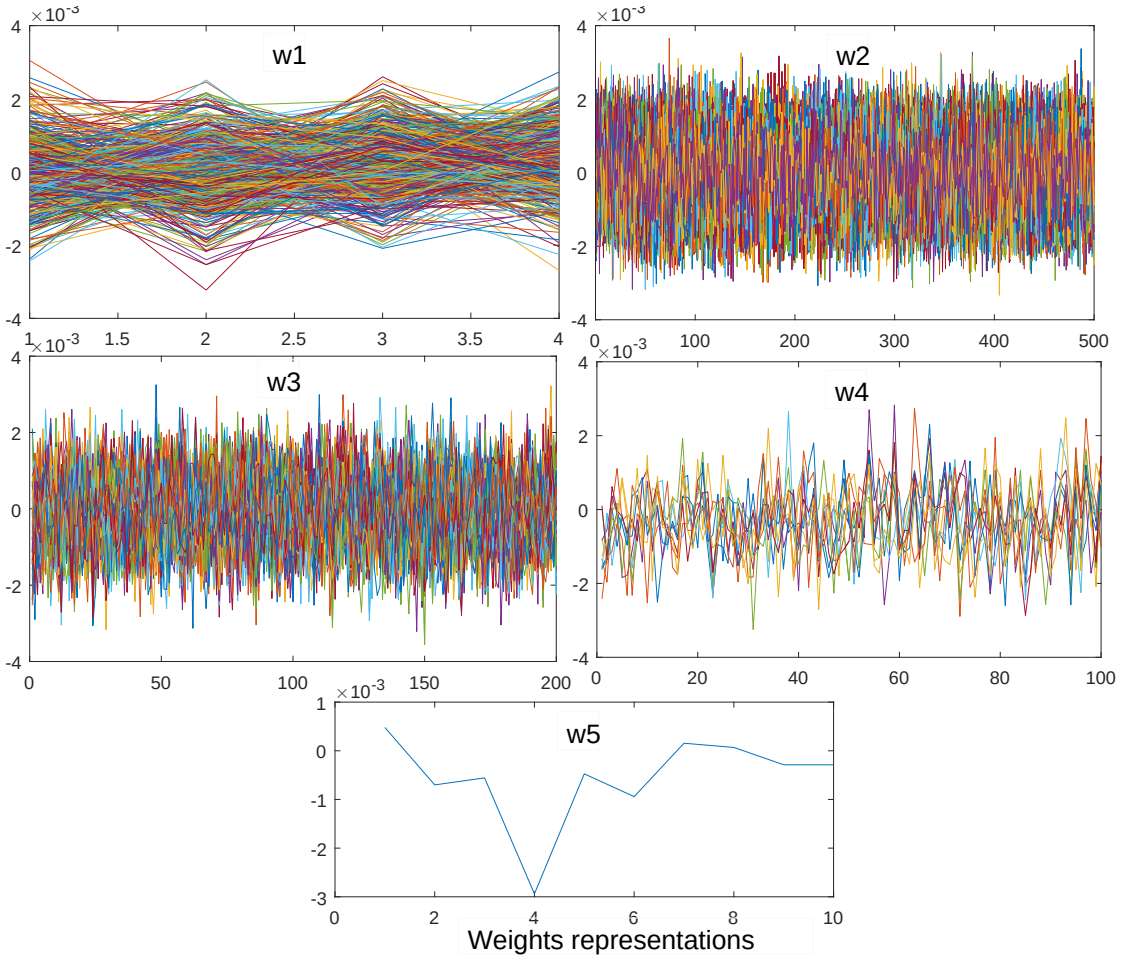

Fig. 4.4 Weights representations of pretrained DBN

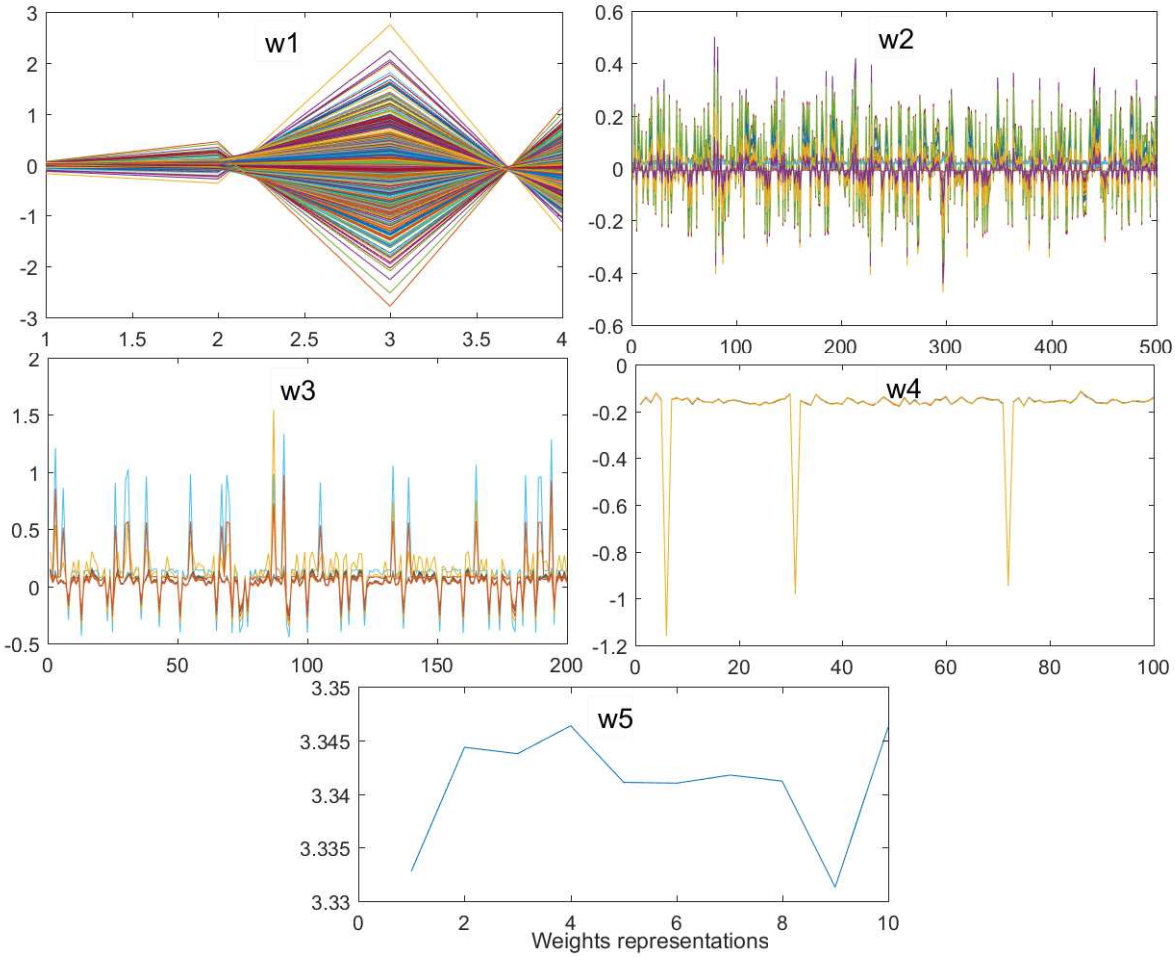

Fig. 4.5 Weights representations after fine tuning

4.3.2 Nonlinear Autoregressive models for multistep forecasting

The task of forecasting temperature on the extracted features from the deep learning architecture was originally achieved by configuring the Non-linear AutoRegressive model with exogenous inputs i.e. NARX ANN. later on, this was eventually concluded with much improved forecasting model by confronting NARMAX ANN.

The literature shows that NARX networks are often much better at discovering long time dependencies than the conventional recurrent neural networks. The NARX feed forward network is created with one hidden layer of 35 neurons and an output layer consisting of one neuron. The size of the hidden layer was selected on the basis of optimal solution given by different models trained in the range of 10 to 40 neurons. The hidden layer activations were processed with a hyperbolic tangent sigmoid transfer function as shown in equation 4.1. The output layer was configured with a linear transfer function.

$$tangsig(x) = \frac{2}{1 + exp(-2 * x) - 1} \tag{4.1}$$

The external input set to train the model contained hour, month samples and aggregated learned features from DBN. The preprocessed input was inserted in the network with tapped delay lines to store the previous values of lagged terms for network training. The forecasting model is shown in Figure 4.6. Subsequently, once the model is trained to capture the data generation patterns, the model is extrapolated to forecast future values. For forecasting along with the external input another input set is considered, i.e., feedback connection from the network output. This indicates, for predicting multistep samples the predictions of y(t) will be used in place of actual future values of y(t). The architectural view of feedback model is represented in Figure 4.7. The number of previous terms to be used by forecaster can be called as delay terms. The associated delay terms were calculated with Autocorrelation. It describes the correlation between the values of the process at different times, as a function of the two times or of the time lag. By use of autocorrelation the sliding window size of lagged terms is calculated as 4. The data set was divided into training, validation and test set to a ratio of 70, 15 and 15. The network was further trained with Levenberg-Marquardt algorithm with 1000 iterations. The performance of the model was measured by Root Mean Square Error (RMSE) and Mean Square Error (MSE) on training, testing and validation datasets.

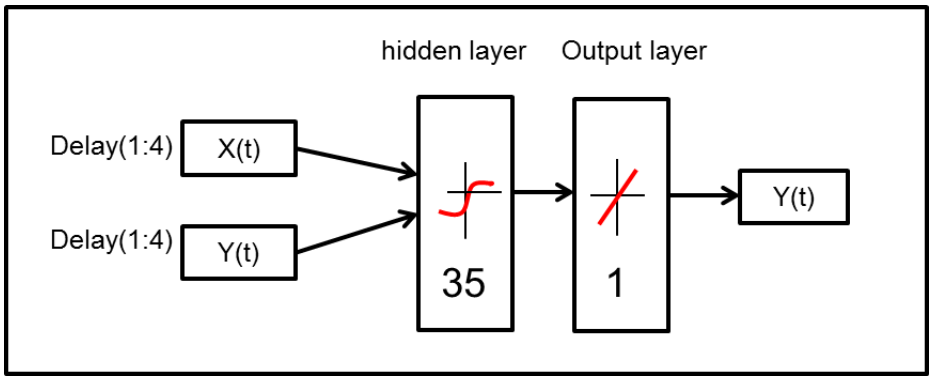

Fig. 4.6 ANN model trained on features extracted from DBN.

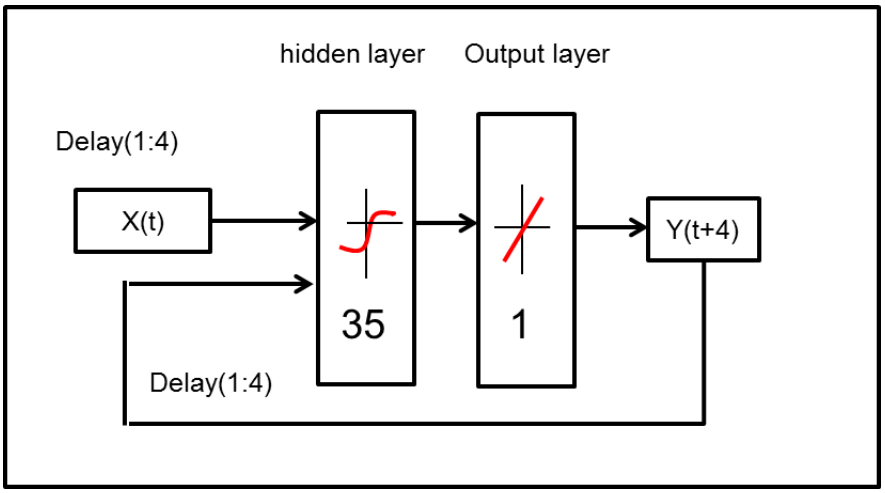

Fig. 4.7 Close-loop ANN model for forecasting future samples.

4.4 Results and Analysis

A useful representations as features are those one, which are significant as an input to a supervised predictor. Henceforth, our model efficiently learned the expressive representations as a feature set for forecasting model. It has the capability to capture possible input configurations which were impossible to identify statistically or mathematically. Figure 4.8 presents the illustration of extracted features from each layer.

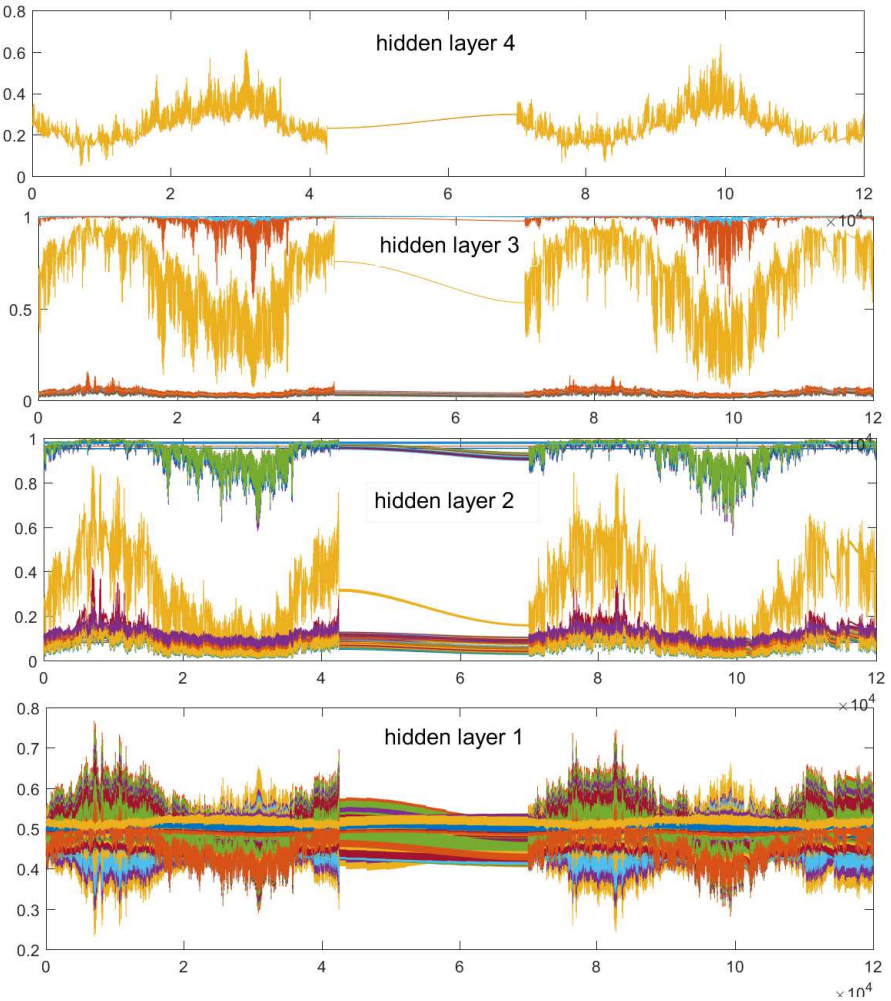

Fig. 4.8 Learnt features extracted from hidden layers of DBN

In the above shown figure, the graph at the top, presents the abstractions of the top most hidden layer close to output layer. Whereas, the graph at the very bottom, shows

the nonlinear relations learned by the first low-level hidden layer of DBN-DNN. The first hidden layer abstractions are low level attributes that reside in time series of temperature. The top level hidden layer was able to learn the representations as close as possible to the target series which is clearly visible from the figure. Apart from this the top hidden layer extracts 10 valuable features in response to the dimension of the layer. Along with the property of being significant, the features in the top layer also contained the element of redundancy. Moreover, the features extracted from the top most hidden layer are sent to forecaster as an input set for future predictions. These representations gave evidence that they are salient for forecasting the future intervals. Several attempts were made to select the appropriate top hidden layer, with the different number of neurons in the range of 1-10. The finest representations learned were with 10 neurons.

The performance measurements of the forecasting model for predictions resulted as 0.0010 RMSE on training, 0.0011 on validation and 0.0011 on the test set. After training of the model, it is further extrapolated to forecast next four steps of time interval. As a forecaster for next hour prediction, it provides 0.0068 error performance on the training data while it gives good performance on the test set by achieving 0.0080 as MSE. Figure 4.9 shows only 50 forecasted samples in order to better highlight the difference between actual and forecasted temperature values. Due to accurate predictions, the actual and forecasted values are almost same.

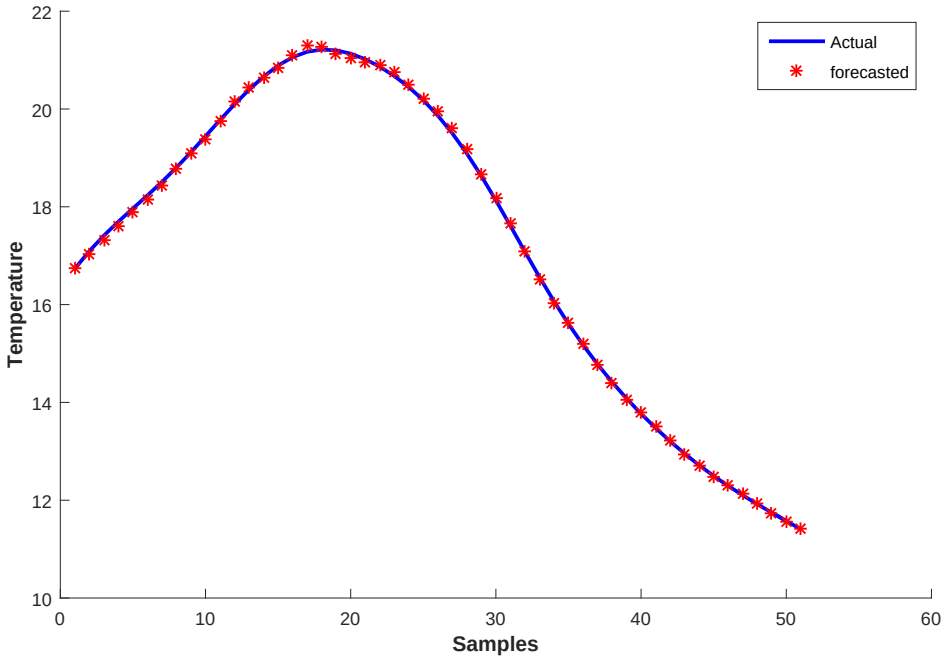

Fig. 4.9 Fifty samples of 4 steps ahead forecasted Temperature.

There is very small set of dispersion between the two presented curves in the above mentioned figure. This indicates that the forecasted temperature drifts a little at peak values. However, the remarkable closeness is visible between the actual and forecasted. This can be evidenced further by reporting statistical analysis for one hour, i.e, for next four samples, forecasting in the form of error histogram and regression plot. The error histogram is shown in figure 4.10. The regression plot which shows correlation is presented in the Figure 4.11. Temperature samples from the test data for next one hour forecasting is shown in Fig. 13. In Fig. 14, only 50 samples are reported to better highlight the difference between actual and forecasted temperature values.

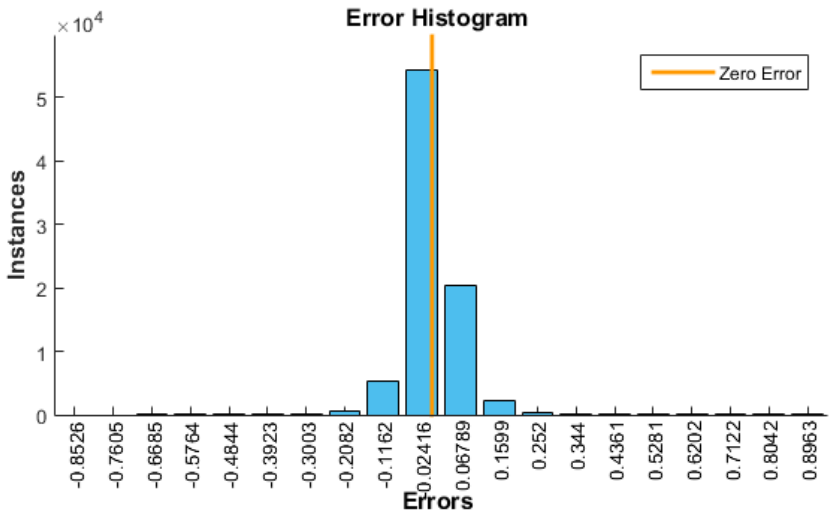

Fig. 4.10 Error Histogram for next 1 hour, 4-steps ahead forecast.

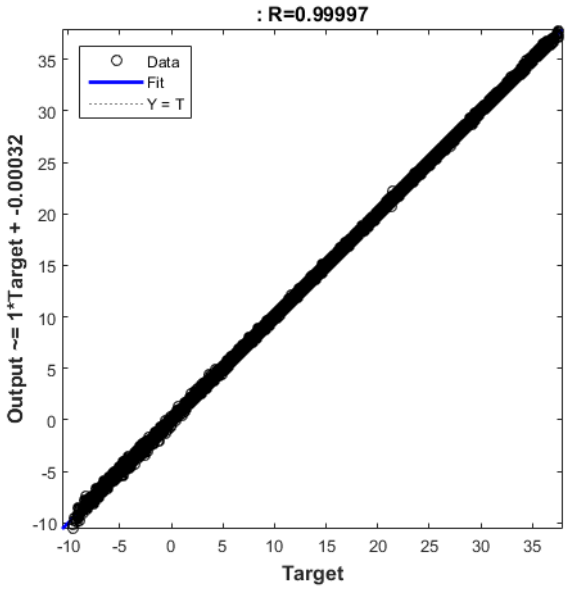

Fig. 4.11 Regression plot for next 1 hour, 4-step ahead forecast.

In order to further validate our proposed approach, we conducted some more experiments with other conventional models. Therefore, for this purpose, three more models of ANN, such as, NAR, NARX and MLP were taken into consideration. The models were developed and trained as predictive models. These models were

trained with the same training set as our proposed model was initially trained. The aforementioned models were trained with a single hidden layer comprising of 35 nonlinear neurons. These neurons were activated by tangent hyperbolic function in a way similar to our proposed Hybrid model. The summarized performance measures of each model are presented in Table 4.3 in order to conduct the relative comparisons.

Table 4.3 Error rate Comparisons with Conventional Models

Models	**Training Error Rate**	**Test Error Rate**
NARMAX	0.0045	0.0040
DBN-DNN-NARX	0.0068	0.0080
NAR	0.0056	0.012
NARX	0.069	0.019
MLP	0.045	0.045

The rate of training error and test error for the performance of each model is measured in MSE. It can be easily seen easily seen from the table 4.3that the proposed hybrid model, i.e, DBN-DNN-NARX noticeably outperforms the other models in the form of test error except of one model, i.e, NARMAX . Although, as far as the training performance of these models is concerned the NARMAX model showed better performance with reduced error rate on both training and test error. Albeit, our model also evidenced to be more appropriate as far as the generalizing capability is concerned. However, the NARMAX achieves the best performance on the test data set as compared to the rest of models.

Literature suggests that the predictive power of these Nonlinear Autoregressive models can be increased when previous errors are incorporated as regressors [120, 121]. Due to this fact, number of lagged windows with different size specifications were considered in the experiments. The above results show that NARMAX models appear to be better predictor. Because, apart from considering the previous lagged terms and external variables they use information about past errors which results in improved prediction performance. MLP model was found to be with the least performance accuracy in the domain of time series forecasting applications as compared to the other trained models in our study. The findings elaborated in Table 4.3 once again justifies the critical importance of deep layered networks and representations learned by these models.

Due to the outstanding performance by NARMAX model, we were encouraged to address it as forecasting model over the representations learned through proposed DBN-DNN. The measure statistics of three top models with least possible error rate are further demonstrated in Table 4.4. The DBN-DNN NARMAX improved over both standalone NARMAX and proposed DBN-DNN NARX forecatser.

Table 4.4 MSE for top three best multistep forecasters

MODELS	ERROR VALUE (MSE)	
	Training Error	Test Error
DBN-DNN NARMAX	0.0030	0.0031
NARMAX	0.0045	0.0040
DBN-DNN NARX	0.0068	0.0080

4.5 Conclusion

We proposed a novel approach for time series forecasting as presented in this chapter. A hybrid approach is proposed with the aim of improving the prediction accuracy. The main innovation is that the extracted feature set, used for temperature forecasting, is not constructed through statistical feature engineering methods, but it is extracted through the deep learning of a Deep Belief Network. Firstly, the nonlinear hierarchical representations are extracted through the hidden layers of the developed DBN-DNN model. Subsequently, the raw input series are transformed into gradually higher level of representations as learnt features.These represent more abstract functions of input at the upper level of the layers by implementing a DBN-DNN architecture. The features extracted learnt the complex mapping between input and output, which is observable from the performance of DBN-DNN. The feature extracted are further, used as data to train the forecaster i.e NARX ANN model. The extracted abstractions reinforced the forecaster ANN model to more accurately predict the temperature, achieving outstanding performance against the mentioned approaches. The results obtained over 5 years of data collection demonstrated that the proposed approach is promising and can be further applied to the prediction of a different set of weather parameters. Our originally proposed model did showed improved performance over some conventional approaches. Albeit, it's performance was not better than NARMAX model. The trained NARMAX model for 4 steps

ahead temperature prediction achieved twice times better accuracy than proposed Hybrid approach. Thus, later on we addressed DBN-DNN NARMAX forecaster providing improved accuracy as compared to rest of the all models.

Chapter 5

EEG signal classification for eye state Identification using SAE

5.1 Introduction

Brain-Computer Interface (BCI) is focusing on the development of communication tools for patients with motor disabilities and providing those individuals with the new modes of communication and control by modulating the brain waves. Apart from this, BCI technology can also reveal valuable information about user state in safety-critical applications, such as driving, industrial environments and security surveillance. BCI is an unambiguous connection path between an external device and stimulated brain.

The electrical activity of the brain is recorded through Electroencephalography (EEG) in the form a signal. It can be further explained as the recording of the brain's spontaneous electrical activity over a period of time. The EEG signal is extracted from several electrodes applied to the scalp's surface. The position and number of electrodes depend on the specific goal of a research. The signal obtained from EEG electrode is usually processed in a separate channel and subsequently amplified. Therefore, one may alternatively use the term channel or electrode [122].

BCI credentials are broadly classified in three categories invasive, partially invasive and noninvasive. The invasive BCIs are implanted directly into the brain using the procedure of neurosurgery. Those BCI devices that are implanted inside the skull, but rest outside the brain comes in the category of partially invasive BCIs.

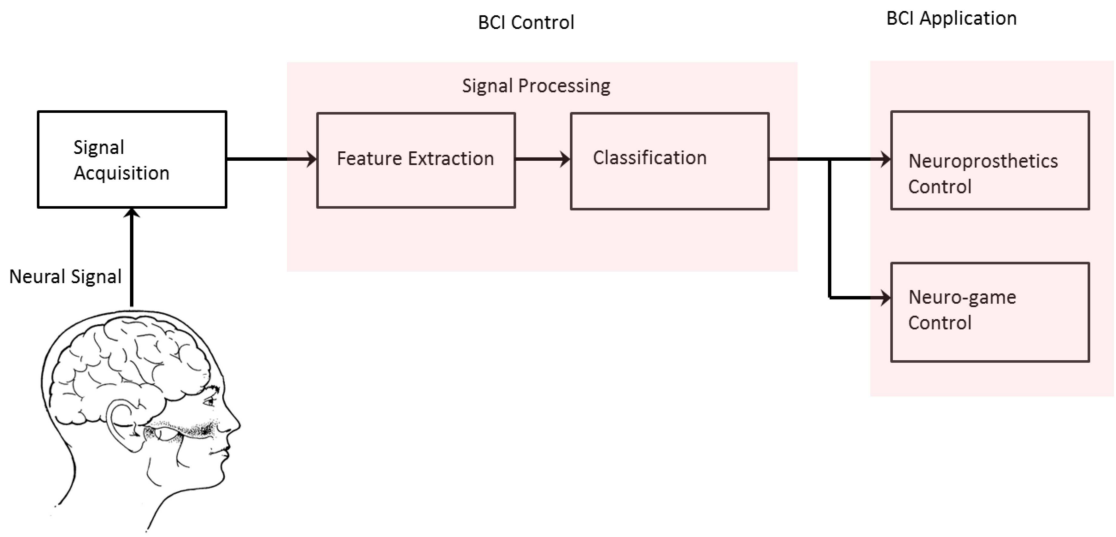

Fig. 5.1 Flow of BCI System

These devices produce better resolution signals than non-invasive BCIs for instance EEG, Magnetoencephalography (MEG) and Magnetic resonance imaging (MRI), where the bone tissue deflects and deforms the signals. The majority of published BCI work involves noninvasive based BCIs. EEG based interfaces may have reduced the spatial resolution and may not efficiently use higher-frequency signals as the skull weakens the signals. However, these interfaces are easy to wear and do not require any surgical procedure.

A basic workflow of BCI system can be seen in Figure 5.1. Once the information is acquired from human brain, then BCI control section recognizes and translates the specific brain signal patterns into device control command through the signal-processing algorithm. Feature extraction can possibly include one or more than one of the following techniques as Autoregressive, Wavelets, Fourier transforms, Laplace filters, Common spatial filters and many others [123–125].

5.2 Problem Statement

Eye state classification based on EEG signal is a kind of common time series prediction problem. Eye state classification can be beneficial for perceiving human cognitive states, which are not only critical to medical care but also significant for everyday life chores. There are various fields where eye State detection plays a

major role for example, infant sleep-waking state identification, driving drowsiness detection, epileptic seizure detection, stress feature identification. In addition to this further implication of this research will be useful for obtaining brain stimuli as an input mode for computer games and to obtain emotions in order to create a virtual Robotic environment. It can also be useful for handicapped persons to control their surrounding electronic control devices.

The deep learning architectures are becoming as essential member of the machine learning family due to its property to learn hierarchical, non linear representations of data. This chapter presents the use of deep learning techniques to classify the eye state: whether open or closed from EEG signals and further demonstrate the improved performance accuracy over previous related state of the art research . The DBN and SAE are preferred and implemented as classifiers. Consequently, both models are further compared in terms of their behavior and performance.

5.3 State of the Art

In the literature, researchers have attempted to successfully remove eye blink artifacts by developing several methods for instance [126] [127] [128]. The authors in [129] provided the comparison of SVM and ANN for classification of EEG eye states such as, eye blink, eye open and eye closed. The SVM was justified as the preferred choice of model over ANN on the basis of performance accuracy given by both models. In addition, a hierarchical classification algorithm is developed using a thresholding method for offline recognition of four directions of eye movements from EEG signals [130]. The researchers in [131] proposed an EEG based eye tracking solution by using the information from two different sources, i.e., Head mounted Video-Oculuography and 16-channeled EEG signal. Their proposed model achieved the accuracy of 97.57% by extracting the features from source by using ICA rather than band-pass preprocessed EEG signals. A novel machine learning approach, incremental attribute learning (IAL) is proposed in [132] for EEG eye state time series classification. The IAL algorithm progressively imports and trains features one at a time to predict the class labeling.

In [133], the researchers have studied the differences among three states, i.e. Eye-closed, Eye-open and Attension states of human using EEG signals. The major goal of their research was to address the issue of how spatial-temporal properties

of alpha rhymes were affected by the change of human brain state. However, the authors did not pursue further towards the predictive model for the classification task of the above mentioned states.

The researchers in [134] established EEG Eye state corpus data set. This dataset based on EEG recordings, is now considered as benchmark data set [135]. In their study, they tested 42 different classifiers for identifying whether an individual's eye is in open state or close state. The best classifier achieved the accuracy of 97.3%.

The same benchmark problem was studied in [136] by using an IAL approach with the Feature ordering based on Accumulative Discriminability (AD). Experimental results show that, with proper feature extraction and feature ordering, IAL can cope with time series classification problems. However, the classification accuracy of their proposed approach did not provide any progress over [134].

The researchers in [137] demonstrated the use of three different ensemble learning approaches on the benchmark problem of EEG based eye state detection. The profound results were obtained by ensembling the Regularized Random Forest RRF and K*. The model achieved classification accuracy of 97.4%.

In [138] novel approach was proposed based on small Neuro-Fuzzy rules. The best possible results were achieved with the average accuracy of 96% and error rate around 4.0%.

In [139] the EEG eye state identification corpus [135] was used to find selected subset features. Incremental Feature Reordering was implemented as algorithm to compute non dominant feature (MND) for Electroencephalography. The removal of MND features gave optimum subset features. This increased the classifier accuracy and efficiency as compared with the traditional method. However, the best possible accuracy achieved by incorporating the mentioned approach is around 95.26%.

5.4 Materials and Method

In this chapter, DLA models are developed and trained as classification models for EEG Eye state identification. To train these deep architectures the learning algorithm for the neural network training is in line with [77, 140]. The flow of steps taken for this work is revealed in Figure 5.2 . The EEG raw data set is pre-processed for noise removal and outlier detection. Feature Extraction is performed subsequently

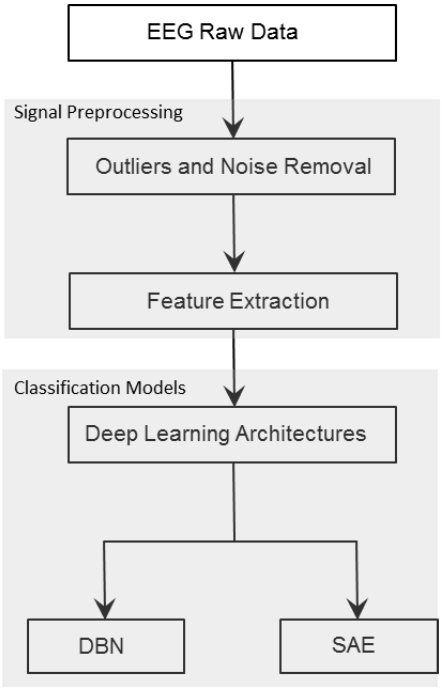

Fig. 5.2 Flow of Research Methodology

as the next step which is further elaborated in the next section. Prior to offering the extracted feature set to the classification model for training, the normalization was performed to scale the signals in the range of (0,1). This is usually done, so that the network learns efficiently. As a final point, the preprocessed dataset was used for the training of classification models. The detail explanation of the steps followed are specified below.

5.4.1 Data Description

A benchmark set for Eye state classification problem was taken from the UCI machine Learning repository. In the corpus, the EEG signal was gathered from 14 different electrodes. The positions of electrode sensors are exposed in Figure 5.3. The eye state either open or closed was captured manually by means of a camera during the EEG measurement and the captured frames were interpreted as 1 and 0 in correspond to Eye-closed and Eye-open states as presented in Figure 5.4. The total duration of the measurement was 117 seconds with 14980 samples.The further details regarding the data set are available in [134].

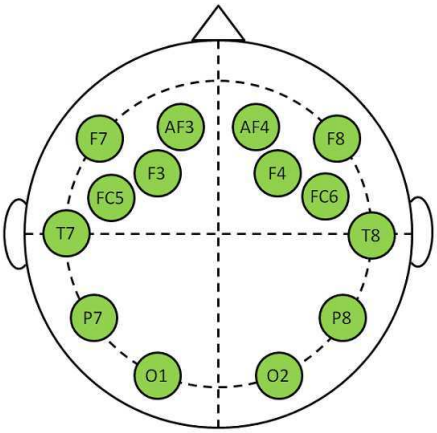

Fig. 5.3 Emotive EEG Neuroheadset set Electrodes position

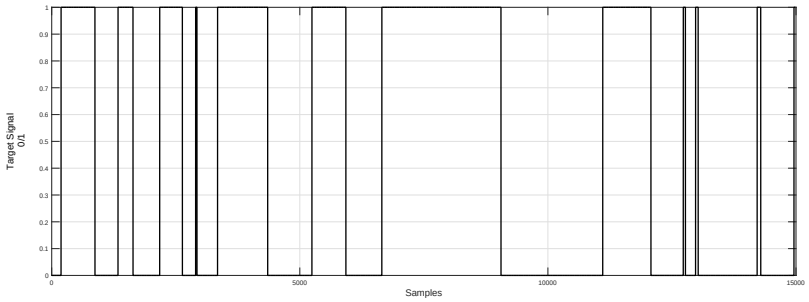

Fig. 5.4 Eye Open and Eye Close states

5.4.2 Preprocessing and Feature Extraction

Imperative selection of appropriate features is foremost important in EEG classification task. It is known fact that EEG signals are non stationary in nature and are usually gathered from multiple channels. Therefore, significant features must be alleviated over the problem, " the curse of dimensionality " and are figured in a time-varying manner. Provision to this, can be various time–frequency diverse formulations. The Discrete Wavelet Transform (DWT) has been widely used for time-frequency analysis in the area of biomedical signals. Therefore, in order to extract expedient features from EEG time series signal DWT was applied.

The DWT decompositions of signals are achieved through high pass and low pass filtering. The approximations are the high-scale, low-frequency components of the signal. The details are the low-scale, high-frequency components. As it visible in Figure 5.5, the EEG signal is further decomposed into level 8 Approximations a8 and Details d1-d8. This level of decomposition is based on the principal frequency components available in the signal. Figure 5.6 indicates the occurrences of 4 level DWT decomposition tree.

Separate 8-level decompositions were computed for a signal residing in each of 14th channel. Subsequently, the frequency bands alpha, beta, gamma, theta and delta are estimated from those computed decompositions. In this way, each signal from each channel is presented in terms of the above mentioned five bands. Therefore the feature set consists of total 14 multiplied by 5 as total attributes. Finally, normalization was performed on the feature set to scale the values in the range of 0 and 1.

5.4.3 Classification Models

In this section, the DLA's are presented as classifier models. Various experiments are demonstrated to analyze and explore the DLAs that falls in two broad and major categories i.e. DBN and SAE. Additionaly, the performance of these models is further investigated on the basis of comparative analysis.

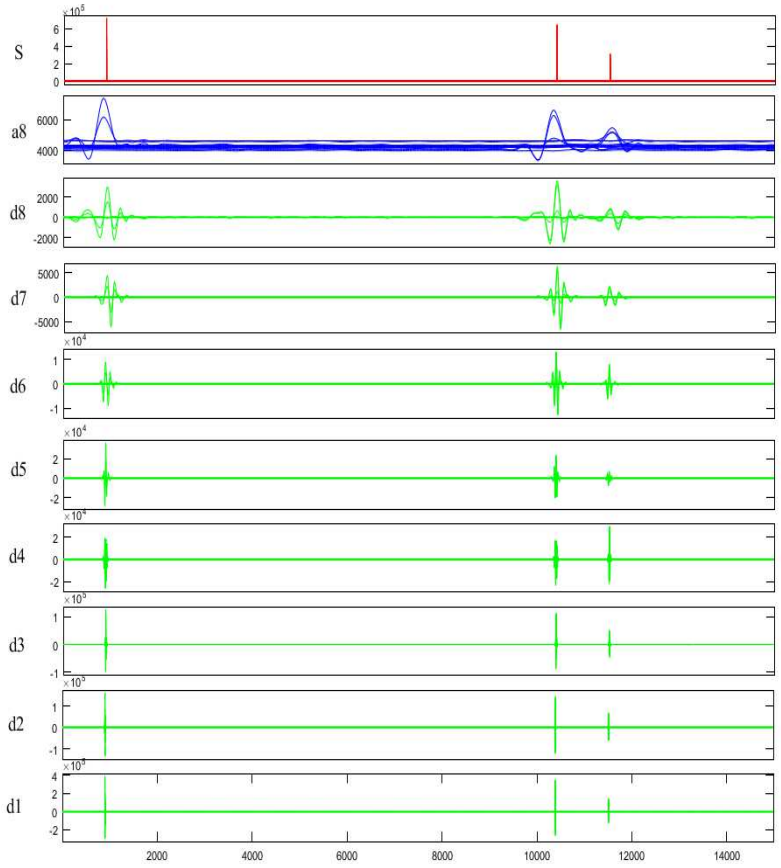

Fig. 5.5 Eight level DWT decompositions of EEG signals

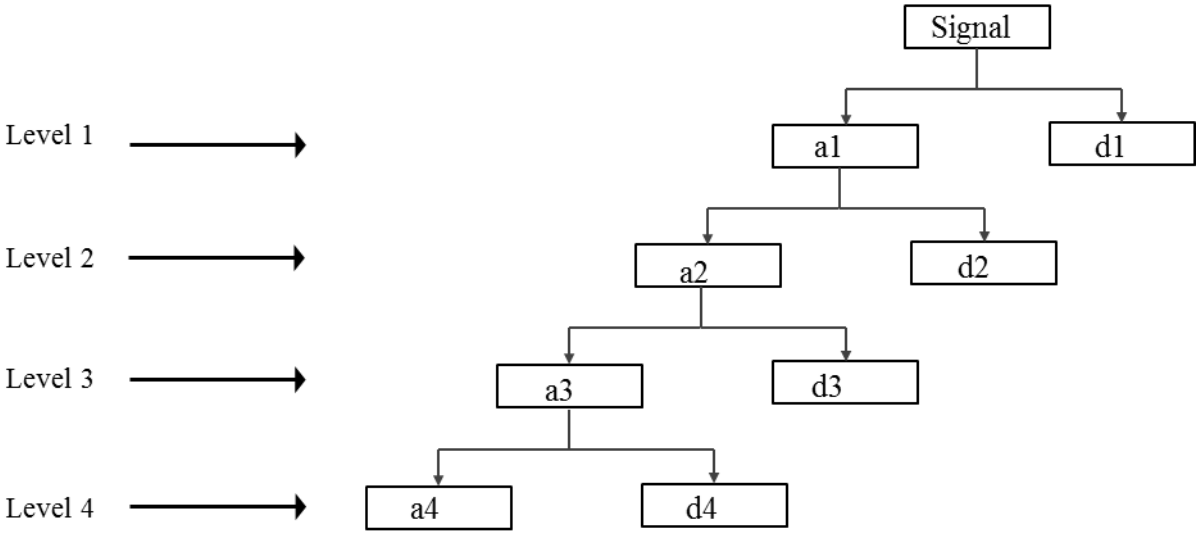

Fig. 5.6 Presentation of Four-level DWT decomposition tree

Table 5.1 Parameter Settings for SAE models

SAE models	Architecture	Hidden layer parameters				Output layer
		β	λ	Sparsity	Activation function	Activation function
SAE 1	70-100-200-2	3	0.001	0.05	Encoder= logsig Decoder= purelin	Softmax
SAE 2	70-200-100-2	3	0.001	0.05		
SAE 3	70-500-200-2	3	0.001	0.05		
SAE 4	70-600-200-2	3	0.001	0.05		
SAE 5	70-100-100-2	1	0.01	0.05		
SAE 6	70-200-100-2	4	0.001	0.06		

SAE Experimental Setup

SAE are available in various forms of variations. For current study, Sparse AutoEncoders were selected, stacked and trained as deeper models for EEG based eye state identification. Several numbers of models comprising on different architecture and parameter settings were developed as sparse classifiers. A number of things are needed to be considered while deciding dimensions for hidden layers of AutoEncoders. A neural network will not be able to learn the underlying structure and pattern that exists in input, incase if the hidden layers do not hold sufficient dimensions. Similarly, if the hidden layers are allowed too much capacity the same problem persists. The possible solution to resolve this is implementing Regularized SAE. Instead of constraining the model capacity or layer dimensions, one way is to use Regularized Autoencoders. Regularized Autoencoder is a sparse model which induces sparsity penalty on loss function and uses it for network training. The loss function for training Regularized AE is presented in equation 5.1

$$JSAE = \sum L\left(v, g_{\theta'}\left(f_{\theta}\left(v\right)\right)\right) + \left(\lambda * L2Regulaization\right) + \left(\beta * SparsityRegularization\right) \tag{5.1}$$

Our concern was to choose the models with the best classification performance on test set, therefore only selected SAE models are included in this chapter for further discussion. Table 5.1 summarises the selected SAE models with their parameter settings. The sparsity proportion is a parameter of sparsity regulerizer which must be selected in the range of 0 to 1, whereas λ and β are the coefficients to L2 weight

regularization and sparsity regularization. Detailed discussion on model accuracy and performance is provided in the Section.

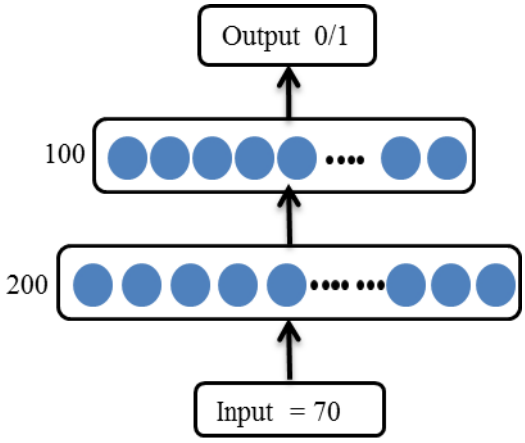

Fig. 5.7 SAE 2 model for classification

DBN Experimental Setup

The second approach was to establish and train DBN model. The model was developed by stacking the RBMs. Initially, in the domain of unsupervised learning Boltzmann machines were introduced, with the aim of learning, distributed internal representations in the hidden layers. A DBN was developed with an input layer consisting of 70 units, three hidden layers of size (600-400-100) and an output layer with one unit to predict the class of an EEG signal i.e. open or close. The ultimate architecture of the trained DBN model is shown in Figure 12. To select the topological structure of the hidden layers of DBN the demonstrations given in [34] were followed. Each layer of the DBN is independently trained as an RBM with the Sigmoid activation function. The states of hidden nodes determined by trained RBM were used as input to the next layer. After unsupervised training, the labels of EEG Eye events were provided at the output layer for linear mapping. The preferred parameter selections for the pretraining of DBN are exposed in Table 5.2.

Once the layers of RBM in DBN were trained, an output layer with Cross Entropy cost function was added. In this way, the network was globally fine tuned with a supervised algorithm to predict the targets. The total 600 iterations were implemented to train the network with the added linear output layer.

Table 5.2 Parameter Settings for Pretraining of DBN model

Parameter	Value
No. of Iterations	10
Learning rate	0.01
Batch size	2
Transfer function	sigmoid

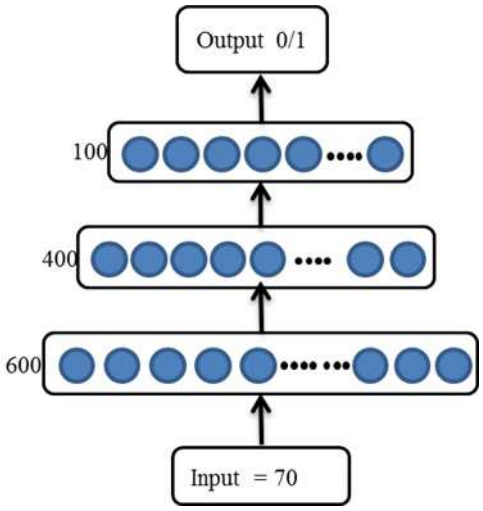

Fig. 5.8 DBN model for classification

5.5 Results and Discussion

The emphasis of the current study was to implement EEG based eye state classification task using two of the several paradigms of Deep Neural Network. This resulted as an improvement over the previous study as discussed later in this section by achieving state of the art performance. In this work, the DBN and SAE were preferred cardinal models as deep learning architectures. The results achieved through Deeper architecture are encouraging and proved to be better in most of the cases than the earlier results reported for EEG eye state classification problem.

In order to study the distributional characteristics boxplot is drawn separately for each class of 14-channelled EEG signal as presented in Figures 5.9 and 5.10 . The boxplot provides a way to envision the range and other statistical characteristics of response variables. As it can be seen in the figures that there is a higher value of difference when the signal amplitude from certain sensors is compared.

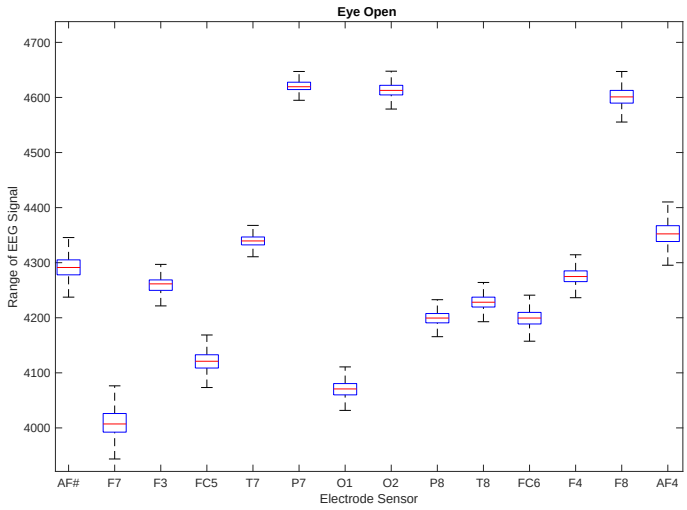

Fig. 5.9 Statistical Range of Electrode signals for eye open state

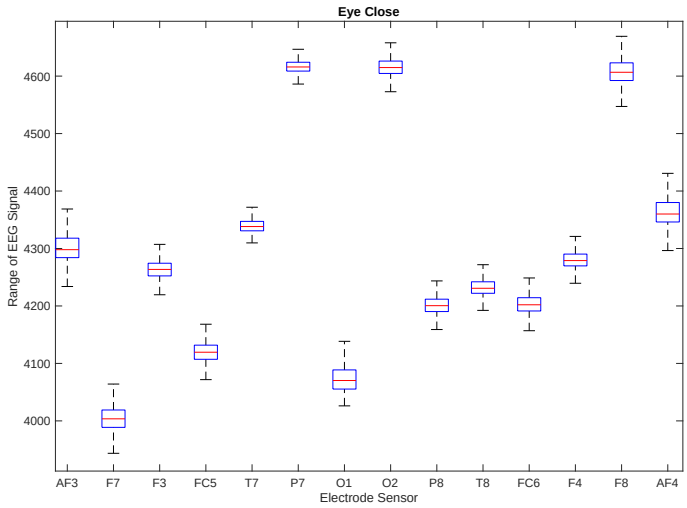

Fig. 5.10 Statistical Range of Electrode signals for eye closed state

Due to strong unsupervised learning phase the deep learning algorithm comparitively reduces the error on test sets than the training set. Apart from this, unsupervised pretraining also behaves as a kind of data dependent regularizer to avoid overfitting criteria.

The unsupervised pretraining being added as local layerwise training, extracts the nonlinear structure present in the data and the salient information from the input

Table 5.3 Performance Metrics of Trained models on Test set

Deep Architectures	Specitivity	Sensitivity	Accuracy	Error rate
DBN	0.99	0.89	97.1%	2.9%
SAE 1	0.95	0.98	97.6%	2.4%
SAE 2	0.96	0.99	98.9%	1.1%
SAE 3	0.94	0.99	97.8%	2.2%
SAE 4	0.94	0.99	97.8%	2.2%
SAE 5	0.80	0.97	92.9%	7.1%
SAE 6	0.74	0.99	91.9%	8.8%

distribution. The elucidation attained in this study are consistent and correlated with the earlier studies conducted in the domain of deeper NN architectures. In the interest of considering the best deep learning model among the implemented and trained as classifier for EEG eye state classification task the comparative analysis is performed. This analysis is based on the performance measuring parameters as summarized in table 5.3. These performance measuring metrices are computed on test samples which were not used in training procedure. The performance of each classifier is calculated using the most commonly used parameters are accuracy, sensitivity and specificity.

The above mentioned table demonstrates that the performance of implemented models. It is almost compatible in terms of prediction accuracy except of SAE 5 and SAE 6. One of the outstanding SAE models i.e SAE 2 tends to afford the accuracy of 98.9%, which is evidently higher than the rest of the other models performance. It seems that the applied approach for SAE 2 exceeds the DBN model in performance with the difference of 1.1% in accuracy . The constraints applied to the optimization procedure of SAE training enforced the model to capture not only the input to target dependency, but also the statistical regularities of the input distribution which resulted in prominently better performance than DBN.

The two exceptional cases as mentioned earlier include, model SAE 5 and SAE 6 did not show good credibility as compared to rest of the models. The tuning parameters in these models, consisting of coefficients for sparsity and L2 regularization, are initialized with different value as compared to the other SAE models as presented in Table 5.3. Contextually, this resulted in considerably lower performance as compared to the other developed SAE models. On the other hand, to enforce the classifier to predict well, the selection of sufficient hidden layer

dimension in deeper architecture is another critical point to consider. To avoid electing the hidden layer sizes arbitrarily, the comprehensive strategy to follow is well explained in earlier chapters. In addition to this some effort must be logically done while investigating the several possibilities for the dimension of hidden layers. During the training procedure, it was also found that in the deeper hierarchical structure of the network, when performing the Representation Learning or feature extraction, the model learns the simple concepts at lower layer levels and more abstract concept are learnt by composing them at higher top layers of the network.

Table 5.4 Comparison with Previous Research

Models	**Accuracy**	**Error rate**
Proposed DLA model, SAE 2	98.9%	1.1%
K*+RRF [137]	97.4%	2.6%
K* [134]	97.3%	2.7%
Neuro-Fuzzy[138]	96%	4.0%
IFR[136]	94%	6.0%

Table 5.4 accentuate the accomplishment of our best proposed classifier i.e SAE 2. The researchers in [134], demonstrated the performance analysis of 42 classifiers on the problem of Eye state classification through EEG signal. Among all of those trained classifiers, K* was found to be the most promising one with an accuracy of 97.30%. However, afterwards the authors in [137] extended the aforementioned work by proposing an ensemble classifier with a better performance of around 97.40% accuracy. Meanwhile the researchers trained a Neuro-Fuzzy classifier which gave the performance accuracy around 96%. In contrast to the above mentioned studies, the proposed SAE 2 model trained with the specified procedure and parameter settings outperforms other models, with the obtained error rate of only 1.1% on the test set. Higher accuracy, around 98.9%, is accomplished on the test samples which demonstrates improved performance over [134],[136],[137] and [138]. We hypothesize that there is a possibility of achieving much better results through deep architectures by implementing Denoising and Contrastive Autoencoders. This indicates a direction for possible future work.

5.6 Conclusion

The directed research, focuses on two approaches of DLA for EEG Eye state classification task as eye open or close. Although this has been done earlier in a number of ways with different Machine learning classifiers, albeit our endeavor improves on the classification accuracy by applying deep architectures as classifiers. Several numbers of models were trained for the study and some of them are presented in this article for significant discussion. A straightforward analysis procedure has been conducted on the basis of different performance measurement metrics. Our trained models with the hyperparameter settings and applied hidden layer dimensions achieved striking performances as compared to the existing ones. Most notably, to our knowledge, this is the first study of EEG based eye state classification tasks using deep Neural Network models, i.e., DBN and SAE. The study conducted here can be applied in different application domains of BCI and for the identification of human cognition. Most imperatively where the artifacts generated by eye movements or eye blinks are crucial, there is a need for them to be identified and further removed.

Chapter 6

Multi step Rainfall forecasting using Temporal DBN

In this part of our research, we propose a simple extension to DBN-RBM model which we developed in order to capture temporal dependencies. Additionally, the extended model is now capable to forecast multi-steps ahead, rather than just performing prediction for next one step ahead. The extended model still maintains its most important computational properties, such that, exact inference and efficient approximate learning using contrastive divergence. It is obvious from earlier studies that, the simplest time series models contain no hidden variables. In general terms, the fully observed models depend upon two types of variables: the first one is vector autoregressive and the subsequent one is Nth order Markov model. Inspite of simplicity these models are constrained by their lack of memory [141].

Generally, on broader spectrum multistep forecasting can be computed through two major strategies. The first is recursive approach and the second one is direct approach. In recursive approach, multi step forecasting is handled iteratively. This means a single time series model is developed and each subsequent forecast is estimated using previously computed forecasts. On the other hand, the direct approach establishes a separate time series model for each forecasting horizon and forecasts are estimated directly by implemented models. However, the choice of selection in between of these two strategies involves a trade-off between bias and variance [142].

An important aspect of the forecasting task is also represented by the size of the horizon. If the one-step forecasting of a time series is already a challenging task, performing multi-step forecasting is more difficult [53] because of additional complications, like accumulation of errors, reduced accuracy, and increased uncertainty [58,49].

Conventionally, multistep forecasting has been managed recursively, where a model is setup as one step forecasting model and each forecast is estimated using previous forecasts. In chapter 4, multi-step ahead predictions were computed by using recursive strategy. Nevertheless, we cannot ignore the fact that minimization of 1-step forecast errors is not guaranteed to provide the minimum over h-steps ahead errors. latterly, in this chapter the emphasise is on direct prediction of multistep forecasting, where a separate time series model for each forecasting horizon is considered and forecasts are computed using the observed data samples. Infact, the direct strategy minimizes the h-step ahead errors instead of considering one-step ahead. Huge number of studies comparing recursive and direct forecasting strategies are present in literature, see [3, 16, 143].

In this chapter, another approach of time series forecasting is presented for muti step horizon. This approach is using Rainfall time series data from METEO weather station of Neuronica Laboratory at Politecnico Di Torino. Forecasting in this method was performed by Temporal Deep Beleif Network and the best model is selected from several baseline models.

Rainfall is significant for agriculture, food production plan, water resource management and likewise other natural systems [144]. The variability of rainfall in space and time, however, renders quantitative forecasting of rainfall extremely difficult [145]. The behaviour and structure of rainfall including its distribution in the temporal and spatial dimensions depends on several variables, for instance, humidity, pressure, temperature and possibly wind direction and its speed. Apart from this, a time series of rainfall usually contains local features too, for example, bursts of heavy rain between prolonged low intensity rainfall durations. Infact, these local features are not fixed in a time slot which renders the prediction of occurrence more difficult. Thus, in our knowledge, the rainfall predicttion was more difficult and challenging task as compared to other metrological parameters.

6.1 Rainfall Dataset

The real time rainfall dataset has been taken from METEO weather station in the same way as we have mentioned in Section 3.4.1. In order to compute the accurate forecast, the foremost step was data analysis.

6.2 Filtering and Noise Removal

In order to smoothen the time series rain data and to reduce noisy fluctuations, we applied and tested number of different filters. But before applying any filter, firstly, we tried to detect some outliers residing in our rain dataset as shown in Figure 6.1. Subsequently, we filter the rain data with number of filters, for example, Moving average, savitzkey golay and other low pas filters. The original and filtered rain data is presented in Figure 6.2. However, when we trained our temporal DBN models with these filtered data, we observed that models were capable to learn well with Moving Average and low pass filtered data. Albeit, later on we found that the input data with moving average filter was introducing some delay in estimated rainfall predictions. Therefore, we continued our experiments using low pass filtered data.

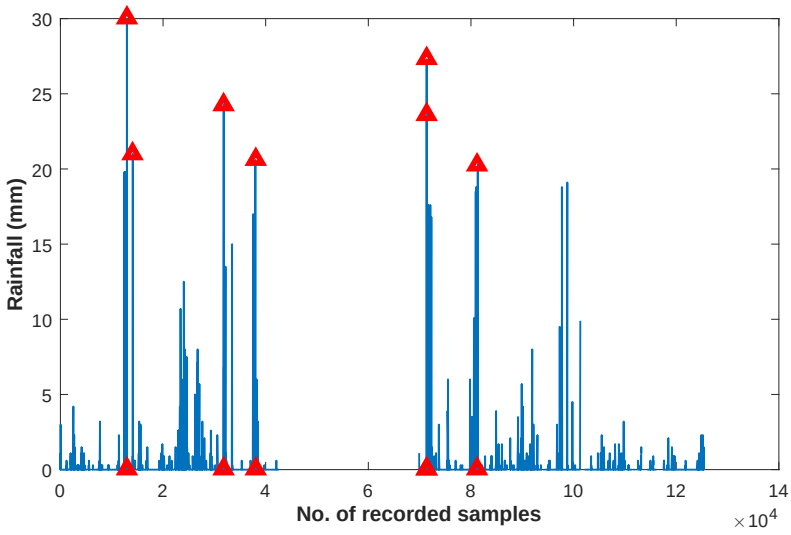

Fig. 6.1 Outliers detection in Rain dataset.

Table 6.1 Mean Square Error on Filtered Rain data.

Filter	MSE
Median	0.0569
Moving Average	0.0447
Low-pass Butterworth	0.0352
Savitzkey golay	0.0302

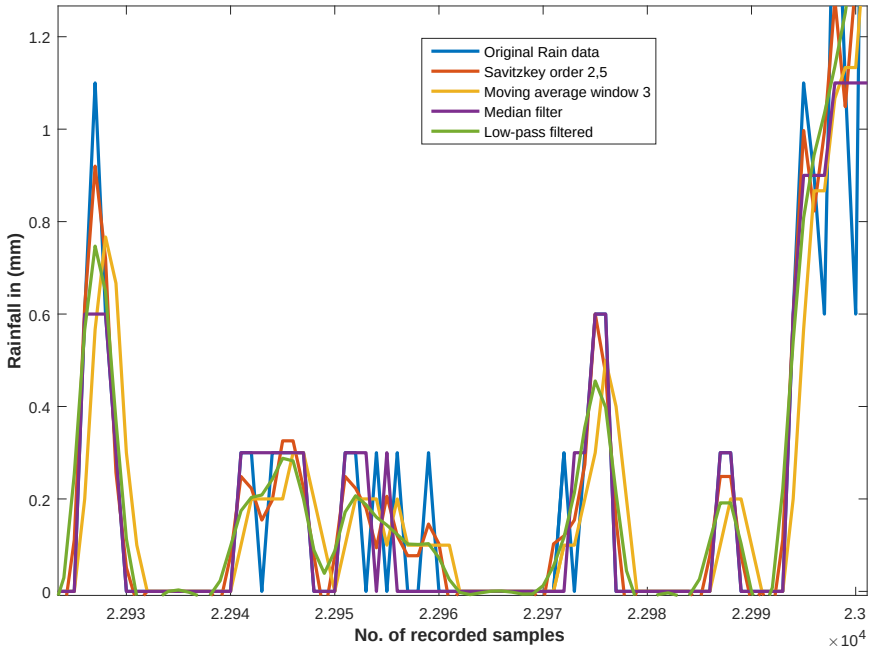

Fig. 6.2 Filtered Rainfall data.

6.3 Feature Extraction

In order to compute accurate rainfall predictions, we must have some meaningful attributes that provides content contribution and possibly reduced error rates. Both internal and external characteristics of rainfall field depend on many factors including pressure temperature, humidity, meteorological characteristics of catchments and so on [146]. However, the rainfall is one of the most difficult variables in the hydrologic cycle. The formation mechanism and forecast of rainfall involve a rather complex physics that has not been completely understood so far [147]. In order to resolve this,

we put some more efforts while creating significantly relevant feature set particularly for rainfall nowcasting.

We attempted with number of different feature sets. Firstly, by adding and deleting the meteorological parameters in sequence. Subsequently, finding the appropriate lagged terms of selected parameters to be used included as features. Successively, we also calculated some statistical features considering mean, standard deviation, variance, maximum, minimum , skewness and kurtosis. We tested our feature sets by training some models and Later on, we found that the DBN models were performing more better if we exclude skewness and kurtosis from selected features. The finalized features to predict rainfall at (t+h) were:

Rain(t+h) = [rain(t), rain(t-1), rain(t-2), rain(t-3), mean(t:t-3), std(t:t-3) humidity(t), pressure(t), temperature(t), humidity(t-1), pressure(t-1), temperature(t-1), humidity(t-2), pressure(t-2), temperature(t-2)]

Where, h is a selected horizon for nowcasting. In our case, it was fixed as 1, 4, and 8 indicating for the next sample, for the next one hour and the next two hours in future.

6.4 Methodology

The RBM models or imitates static category of data and it does not integrate any temporal information by default. In order to model time series data, as it was done in Chapter 3, we added the autoregressive information as input by considering the previous lag terms in series. Here, in this current approach, apart from autoregressive terms, we also incorporate some statistical dependencies from the temporal structure of time series data in the form of input feature set parameters. This was done due to the fact that, multistep forecasting for longer horizon is more challenging task. Therefore, some extra statistical considerations and computations needed to be done in order to understand the proper underlying behaviour of temporal sequence.

The addition tapped delay past inputs introduce temporal dependency and transform the model from static to dynamical form. In general, the dynamic ANN depends on a set of input predictor data. Consequently, the dataset needs to define and represent relevant attributes, to be of good quality and to span comparable period of time with data series [148].

6.5 Results and Discussion

In weather forecasting, specifically rainfall prediction is one of the most imperatives, demanding, critical operational problem. It is complex and difficult because, in the field of meteorology decisions are taken with a degree of uncertainty Consequently, this happens due to chaotic nature of the atmosphere which limits the validity of deterministic forecasts. Generally, the required parameters to predict rainfall are extremely complicated and highly variable. This increases the uncertainty in rainfall prediction task even for the shorter horizons [149].

It needs much more effort to compare and contrast different types of models available so far in literature, in order to evaluate their performance at rainfall forecasting. Because the earlier work done for rainfall forecasting usually provides the comparison of their output with observed values. Thus, this evaluation becomes data-dependent due to the difference of data taken for the different regions and time periods. However, considering the fact that in order to compare our implemented forecasting model, we also trained some significant nonlinear autoregressive neural networks on our data.

For each forecasting horizon, a separate model is trained and optimized. However, as it has been discussed earlier that, multi step forecasting is much more challenging than one-step ahead. Because, as the forecasting horizon is increased, it is obvious that the propagation of error in each sample will be raised. Due to this known fact, the performance accuracy for longer horizon is slightly less than the short forecasting horizon.

While training and selecting the final model for each separate forecasting horizon, multiple models were developed and one for each forecasting horizon was finalized. Additionally, the best model considered was chosen on the basis of performance evaluation. The performance for each model was measured and further compared by using RMSE, MSE and R parameters.

6.5.1 One step ahead forecasting

The commendable deep RBM model for one step ahead rainfall forecasting was chosen with the architecture of (800-400-100-10) hidden layers resulting the depth of four levels. Apart from this, one input layer consisting of fifteen units and one

output layer with one unit for predicting the target. The model performed well with RMSE of 0.0021 on training data and 9.558e-04 on test data set.

6.5.2 Four steps ahead forecasting

In order to perform four steps ahead forecasting the model with the following hidden layer dimensions (600-400-100-10) is proposed. Similar to earlier mentioned model, the input layer is created with 15 nodes and an output layer is connected for predictions. It resulted with RMSE of 0.0093 on training and 0.0057 on test data set.

6.5.3 Eight steps ahead forecasting

Eight step ahead forecasting was more troublesome task than the rest two. Considerably, more networks were attempted to be trained and to be selected as the optimal one. However, the accuracy of each model varies with very slight difference. The outcomes observed for each trained model were almost equivalent. Although, immensely different architectures were optimized. Table 6.2 summarizes the performance measures. The deep architecture of DBN-RBM with (300-200-100-10) as hidden layer dimensions was found to be the most appropriate and adequate for eight steps ahead forecasting.

As forecasting for longer horizon is an arduous task, a deep CNN model is also introduced in this section of our research activity. The latest literature exhibit that the structurally diverse CNN stands out for their pervasive implementation and have led to impressive results [150, 151, 59, 152].

Convolution is a mathematical concept used heavily in Digital Signal Processing when dealing with signals that take the form of a time series. In lay terms, convolution is a mechanism to combine or blend two functions of time in a coherent manner. Thus, the CNN learns the features from the input data. The real values of the kernel matrix change with each learning iteration over the training set, indicating that the network is learning to identify which regions are of significance for extracting features from the data. In CNN model, the convolution filter or kernel is basically an integral component of the layered architecture. The kernels are then convolved with the input volume to obtain so-called 'activation maps'. Activation maps indicate 'activated' regions, i.e. regions where features specific to the kernel have been

detected in the input data. In general, the kernel used for the discrete convolution is small, this means that the network is sparsely connected. This further reduces the runtime inference and backpropagation in the network.

CNN also typically include some kind of spatial pooling in their activation functions. This helps to take summary statistics over small spatial regions of input in order to make the final output invariant to small spatial translations of the input. CNNs have been very successful for commercial image processing applications since early.

According to our objective here, in contrast to image classification the traditional CNN is modified and applied to time series prediction task for eight steps ahead forecasting of rainfall series. The proposed CNN includes four layers as shown in figure 6.3. The first convolutional layer was developed by 3 filters with kernel size of (3,1). Similarly, the second conv layer contained 10 filters with the same size as earlier (3,1). The pooling layer was added by following the "average" approach for sub-sampling. However, in our case the averaging factor was unity. For fully connected layers, tangent hyperbolic activations were used followed by a linear layer for output predictions. Table 6.2 presents the details related with the performance of each model for eight steps ahead forecasting. It is clearly visible from the above mentioned table that the third model with the deep architecture of 300-200-100-10 outplayed than the rest of models. Its performance in terms of RMSE, MSE and R is superior in comparison to other models.

Table 6.2 Performance measures of proposed deep architectures for Eight step ahead rainfall forecasting.

S.No	Hidden layers	RMSE		MSE		R
		Training	Test	Training	Test	
1*	600-400-100-10	0.0187	0.0068	3.4842e-04	4.6692e-05	0.943
2*	500-500-100-10	0.0188	0.0070	3.5372e-04	4.9085e-05	0.941
3*	300-200-100-10	0.0182	0.0068	3.3915e-04	4.5971e-05	0.944
4	800-400-100-10	0.0184	0.0070	3.4021e-04	4.9127e-05	0.944
5	CNN	0.0209	0.0072	4.3475e-04	5.1615e-05	0.929

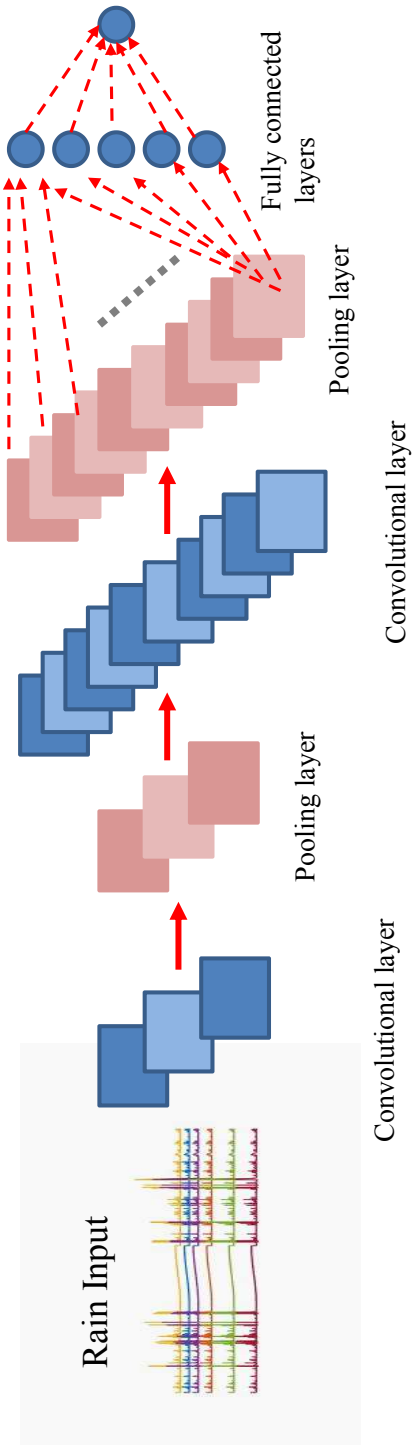

Fig. 6.3 Convolutional Neural Network for rain prediction.

Chapter 7

Conclusions and Future Developments

Recently, Deep learning approaches are invading rapidly and providing remarkable performance in almost all domains. We studied four types of deep learning models over temporal data and further compared it with other models available in the literature. One of our research question was weather the DBN-DNN models are useful for time series prediction or not. And, to further explore that weather these models capable enough to capture temporal dependencies or not. Our observations found that these models can achieve acceptable performance for time series prediction considering some specifications to follow while developing and training these model.

To model the complex time series data, one possibility is to develop more robust and meaningful features that are capable to detain the appropriate information and can accurately represent the behaviour of underlying data. However, it is obvious that computing domain specific features for each task is expensive, time-consuming and needs expertise of the data. Alternative to this, is to use unsupervised feature learning. It is not something new, this choice of representation has an enormous effect on the performance of machine learning algorithms. Deep learning aims to learn good or in other sense highly meaningful representations from low level raw features.

In summary, this thesis presents a collection of the work done on time series forecasting. Particularly, bringing up a minor novelty in earlier deep learning models like, DBN and SAE to work with temporal sequences. Additionally, to produce

accurate estimations not only for 1-step ahead predictions but also multi-steps ahead. The two variations of forecasting strategies, recursive and direct approach both are considered for multi-step ahead forecasting. Necessary data transformations are applied prior to implementing the forecasting models. The effectiveness of our proposed approach was further tested on two benchmark time series.

One more critical aspect that we focused in our thesis is optimizing the hyper-parameters of such deeper networks. The more deeper one goes the more hyper-parameters one gets to be done and it is often difficult to select the adequate architecture. While occupying the much of the available computational resources of Neuronica laboratory and continuously running those experiments, it was observed that the already defined criteria in recent literature for selecting the parameters and hyper-parameters to train either unsupervised pretraining, greedy layer-wise or convolutions only provides a good starting point for conducting the experiments but it is not necessary that it is often validated which leaves an open question. This needs to be answered by solid comparative experimental work and theoretical analysis.

This thesis contributes a little advancement in deep learning algorithms for time series forecasting applications. Although, the research presented in this thesis work has reached its aims, but there are still some limitations and shortcomings. For example, for DBN-DNN architecture we used sigmoid activation functions on hidden units. Interesting observations could be made if variety of hidden units, such as, RBF and ReLUs were investigated . For optimization, we only considered SGD with momentum, we hypothesize that much more accurate results are possible to achieve with adaptive learning rate algorithms. In reality different experts working on different tasks do not always agree on a specific practice of training neural networks. Correspondingly, the paradigms, hypothesis and ideas suggested in our work is limited to narrow domain of time series data. Thus, it can be further extended on wide spectrum by considering the time series data on larger spectrum.

Deep learning is becoming the most exciting field in AI. However, Deep learning algorithms are computational intensive and highly tricky to train. DNN research favours using GPUs. In recent years , due to the advent of deep learning algorithms, there has been a resurgence in the field of machine learning and neural networks. Google, Microsoft, facebook and similar big players are making some impressive acquisitions which is spreading beyond the academic world. Apart from efficient learning models some of the credit goes to the abundance of raw data and to cheap

computational power available via GPUs. FPGAs provide superior energy efficiency compared to high-end GPUs, they are known for offering top peak floating point performance. Although, deep networks have improved accuracy, but greatly increase the number of parameters and model sizes. This increases the number of parameters and model sizes. From Future perspective, the race between GPUs and FPGAs will eventually facilitate Deep learning algorithms.

We also observed in our study that the skills required to trained these models are reduced condition upon the gigantic availability of data. And, this is now what we have, the "Big Data".

For the future development, we envision that much more robust models can be developed for long time series forecasting, if we add the residuals errors as inputs through feedback loop in the DBN-DNN training process.

References

[1] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. In *Business Intelligence*, pages 62–77. Springer, 2013.

[2] George C Tiao and Ruey S Tsay. Some advances in non-linear and adaptive modelling in time-series. *Journal of forecasting*, 13(2):109–131, 1994.

[3] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16):2861–2869, 2007.

[4] Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083, 2012.

[5] Thomas M Mitchell. Machine learning. *New York*, 1997.

[6] Nikolaos Kourentzes and S Crone. Automatic modelling of neural networks for time series prediction-in search of a uniform methodology across varying time frequencies. 2008.

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[8] GEP Box and GM Jenkins. Time series, forecasting and control, rev. ed, 1976.

[9] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[10] Chris Brooks. Univariatetime series modelling and forecasting. *Introductory Econometrics for Finance. 2nd Ed. Cambridge University Press. Cambridge, Massachusetts*, 2008.

[11] John H Cochrane et al. Financial markets and the real economy. *Foundations and Trends® in Finance*, 1(1):1–101, 2005.

[12] Keith W Hipel and A Ian McLeod. *Time series modelling of water resources and environmental systems*, volume 45. Elsevier, 1994.

[13] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010.

[14] Souhaib Ben Taieb, Rob J Hyndman, et al. Recursive and direct multi-step forecasting: the best of both worlds. *Wroking paper*, 2012.

[15] Haibin Cheng, Pang-Ning Tan, Jing Gao, and Jerry Scripps. Multistep-ahead time series prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774. Springer, 2006.

[16] Coşkun Hamzaçebi, Diyar Akay, and Fevzi Kutay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications*, 36(2):3839–3844, 2009.

[17] Douglas M Kline and GP Zhang. Methods for multi-step time series forecasting with neural networks. *Neural networks in business forecasting*, pages 226–250, 2004.

[18] Antti Sorjamaa and Amaury Lendasse. Time series prediction using dirrec strategy. In *ESANN*, volume 6, pages 143–148, 2006.

[19] Gianluca Bontempi. Long term time series prediction with multi-input multi-output local learning. *Proc. 2nd ESTSP*, pages 145–154, 2008.

[20] Gianluca Bontempi and Souhaib Ben Taieb. Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International journal of forecasting*, 27(3):689–699, 2011.

[21] Souhaib Ben Taieb, Gianluca Bontempi, Antti Sorjamaa, and Amaury Lendasse. Long-term prediction of time series by combining direct and mimo strategies. In *2009 International Joint Conference on Neural Networks*, pages 3054–3061. IEEE, 2009.

[22] Nguyen Hoang An and Duong Tuan Anh. Comparison of strategies for multi-step-ahead prediction of time series using neural network. In *Advanced Computing and Applications (ACOMP), 2015 International Conference on*, pages 142–149. IEEE, 2015.

[23] Iebeling Kaastra and Milton S Boyd. Forecasting futures trading volume using neural networks. *Journal of Futures Markets*, 15(8):953–970, 1995.

[24] W-C Chiang, Timothy L Urban, and Gable W Baldridge. A neural network approach to mutual fund net asset value forecasting. *Omega*, 24(2):205–215, 1996.

[25] Radu Drossu and Zoran Obradovic. Regime signaling techniques for non-stationary time series forecasting. In *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*, volume 5, pages 530–538. IEEE, 1997.

[26] C Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, 44(1-2):161–183, 2001.

[27] A Patra, S Das, SN Mishra, and MR Senapati. An adaptive local linear optimized radial basis functional neural network model for financial time series prediction. *Neural Computing and Applications*, 28(1):101–110, 2017.

[28] Luca Mesin, Fiammetta Orione, Riccardo Taormina, and Eros Pasero. A feature selection method for air quality forecasting. In *International Conference on Artificial Neural Networks*, pages 489–494. Springer, 2010.

[29] Sanam Narejo and Eros Gian Alessandro Pasero. Time series forecasting for outdoor temperature using nonlinear autoregressive neural network models. *Journal of Theoretical and Applied Information Technology*, 94, 2016.

[30] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural networks*, 5(6):961–970, 1992.

[31] U Brunelli, VINCENZO Piazza, L Pignato, F Sorbello, and S Vitabile. Two-days ahead prediction of daily maximum concentrations of so 2, o 3, pm10, no 2, co in the urban area of palermo, italy. *Atmospheric Environment*, 41(14):2967–2995, 2007.

[32] Sven F Crone, Michele Hibon, and Konstantinos Nikolopoulos. Advances in forecasting with neural networks? empirical evidence from the nn3 competition on time series prediction. *International Journal of Forecasting*, 27(3):635–660, 2011.

[33] Hong Thom Pham, Bo-Suk Yang, et al. A hybrid of nonlinear autoregressive model with exogenous input and autoregressive moving average model for long-term machine state forecasting. *Expert Systems with Applications*, 37(4):3310–3317, 2010.

[34] Muhammad Ardalani-Farsa and Saeed Zolfaghari. Chaotic time series prediction with residual analysis method using hybrid elman–narx neural networks. *Neurocomputing*, 73(13):2540–2553, 2010.

[35] Rohitash Chandra and Mengjie Zhang. Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction. *Neurocomputing*, 86:116–123, 2012.

[36] Soukayna Mouatadid and Jan Adamowski. Using extreme learning machines for short-term urban water demand forecasting. *Urban Water Journal*, pages 1–9, 2016.

[37] Mukesh Tiwari, Jan Adamowski, and Kazimierz Adamowski. Water demand forecasting using extreme learning machines. *Journal of Water and Land Development*, 28(1):37–52, 2016.

[38] Song Li, Lalit Goel, and Peng Wang. An ensemble approach for short-term load forecasting by extreme learning machine. *Applied Energy*, 170:22–29, 2016.

[39] Nargess Hosseinioun. Forecasting outlier occurrence in stock market time series based on wavelet transform and adaptive elm algorithm. *Journal of Mathematical Finance*, 6(01):127, 2016.

[40] Deyun Wang, Shuai Wei, Hongyuan Luo, Chenqiang Yue, and Olivier Grunder. A novel hybrid model for air quality index forecasting based on two-phase decomposition technique and modified extreme learning machine. *Science of The Total Environment*, 2016.

[41] Wei Guo, Tao Xu, and Zonglei Lu. An integrated chaotic time series prediction model based on efficient extreme learning machine and differential evolution. *Neural Computing and Applications*, 27(4):883–898, 2016.

[42] Lean Yu, Wei Dai, and Ling Tang. A novel decomposition ensemble model with extended extreme learning machine for crude oil price forecasting. *Engineering Applications of Artificial Intelligence*, 47:110–121, 2016.

[43] Yongning Zhao, Lin Ye, Zhi Li, Xuri Song, Yansheng Lang, and Jian Su. A novel bidirectional mechanism based on time series model for wind power forecasting. *Applied Energy*, 177:793–803, 2016.

[44] Ömer Faruk Ertugrul. Forecasting electricity load by a novel recurrent extreme learning machines approach. *International Journal of Electrical Power & Energy Systems*, 78:429–435, 2016.

[45] Enzo Busseti, Ian Osband, and Scott Wong. Deep learning for time series modeling. *Technical report, Stanford University*, 2012.

[46] James NK Liu, Yanxing Hu, Jane Jia You, and Pak Wai Chan. Deep neural network based feature representation for weather forecasting. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

[47] Mladen Dalto, Jadranko Matuško, and Mario Vašak. Deep neural networks for ultra-short-term wind forecasting. In *Industrial Technology (ICIT), 2015 IEEE International Conference on*, pages 1657–1663. IEEE, 2015.

[48] Wan He. Deep neural network based load forecast. *Computer Modelling & New Technologies*, 18(3):258–262, 2014.

[49] Takashi Kuremoto, Shinsuke Kimura, Kunikazu Kobayashi, and Masanao Obayashi. Time series forecasting using a deep belief network with restricted boltzmann machines. *Neurocomputing*, 137:47–56, 2014.

[50] Amaury Lendasse Erkki, Erkki Oja, and Olli Simula. Time series prediction competition: the cats benchmark. 2004.

[51] Amaury Lendasse, Erkki Oja, Olli Simula, and Michel Verleysen. Time series prediction competition: The cats benchmark, 2007.

[52] Aditya Grover, Ashish Kapoor, and Eric Horvitz. A deep hybrid model for weather forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 379–386. ACM, 2015.

[53] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[54] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. *Advances in neural information processing systems*, 19:1345, 2007.

[55] Ilya Sutskever and Geoffrey E Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *AISTATS*, volume 2, pages 548–555, 2007.

[56] Nikhil Garg and James Henderson. Temporal restricted boltzmann machines for dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 11–17. Association for Computational Linguistics, 2011.

[57] Ivan Sorokin. Classification factored gated restricted boltzmann machine. In *AALTD@ PKDD/ECML*, 2015.

[58] Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[59] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[60] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[61] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[62] Kai Chen, Mathias Seuret, Marcus Liwicki, Jean Hennebert, and Rolf Ingold. Page segmentation of historical document images with convolutional autoencoders. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1011–1015. IEEE, 2015.

[63] Alireza Makhzani and Brendan Frey. A winner-take-all method for training sparse convolutional autoencoders. In *NIPS Deep Learning Workshop*. Citeseer, 2014.

[64] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

[65] Svitlana Galeshchuk and Sumitra Mukherjee. Deep networks for predicting direction of change in foreign exchange rates. *Intelligent Systems in Accounting, Finance and Management*, 2017.

[66] Yongzhi Qu, Miao He, Jason Deutsch, and David He. Detection of pitting in gears using a deep sparse autoencoder. *Applied Sciences*, 7(5):515, 2017.

[67] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagc. Deep learning techniques in big data analytics. In *Big Data Technologies and Applications*, pages 133–156. Springer, 2016.

[68] Mohammadmehdi Ezzatabadipour, Parth Singh, Melvin D Robinson, Pablo Guillen-Rondon, and Carlos Torres. Deep learning as a tool to predict flow patterns in two-phase flow. *arXiv preprint arXiv:1705.07117*, 2017.

[69] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[70] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[71] Nagwa M Elaraby, Mohammed Elmogy, and Shereif Barakat. Deep learning: Effective tool for big data analytics. *International Journal of Computer Science Engineering (IJCSE)*, 5(05), 2016.

[72] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[73] Haidong Shao, Hongkai Jiang, Xun Zhang, and Maogui Niu. Rolling bearing fault diagnosis using an optimization deep belief network. *Measurement Science and Technology*, 26(11):115002, 2015.

[74] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

[75] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[76] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[77] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[78] Abdel-rahman Mohamed, Tara N Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E Hinton, and Michael A Picheny. Deep belief networks using discriminative features for phone recognition. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5060–5063. IEEE, 2011.

[79] Li Deng, Michael L Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoffrey E Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech*, pages 1692–1695. Citeseer, 2010.

[80] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[81] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 24–29. IEEE, 2011.

[82] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[83] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *ISMIR*, pages 729–734, 2011.

[84] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[85] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155, 2003.

[86] Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.

[87] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *AISTATS*, volume 351, pages 423–424, 2012.

[88] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011.

[89] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[90] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[91] Sanam Narejo and Eros Gian Alessandro Pasero. An application of internet traffic prediction with deep neural network. In *New Perspectives in neural networks / M. Faundez ; A.Esposito ; C. Morabito ; E. Pasero,*, volume 55. Springer.

[92] Sanam Narejo and Eros Gian Alessandro Pasero. Meteonowcasting using deep learning architecture. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE & APPLICATIONS*, 8(8), 2017.

[93] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[94] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, pages 599–619. Springer, 2012.

[95] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

[96] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10(Jan):1–40, 2009.

[97] Huifang Feng and Yantai Shu. Study on network traffic prediction techniques. In *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, volume 2, pages 1041–1044. IEEE, 2005.

[98] Kihong Park and Walter Willinger. *Self-similar network traffic and performance evaluation*. Wiley Online Library, 2000.

[99] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking (TON)*, 9(4):392–403, 2001.

[100] Pengjian Shang, Xuewei Li, and Santi Kamae. Nonlinear analysis of traffic time series at different temporal scales. *Physics Letters A*, 357(4):314–318, 2006.

[101] Pengjian Shang, Xuewei Li, and Santi Kamae. Chaotic analysis of traffic time series. *Chaos, Solitons & Fractals*, 25(1):121–128, 2005.

[102] Z-L Zhang, Vinay J Ribeiro, Sue Moon, and Christophe Diot. Small-time scaling behaviors of internet backbone traffic: an empirical study. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1826–1836. IEEE, 2003.

[103] Chun You and Kavitha Chandra. Time series models for internet data traffic. In *Local Computer Networks, 1999. LCN'99. Conference on*, pages 164–171. IEEE, 1999.

[104] Mikio Hasegawa, Gang Wu, and M Mizuni. Applications of nonlinear prediction methods to the internet traffic. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 3, pages 169–172. IEEE, 2001.

[105] Paulo Cortez, Miguel Rio, Miguel Rocha, and Pedro Sousa. Internet traffic forecasting using neural networks. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 2635–2642. IEEE, 2006.

[106] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 2014.

[107] Mladen Dalto. Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting. In *IEEE International Conference on Industrial Technology*. IEEE, 2015.

[108] Tiago Prado Oliveira, Jamil Salem Barbar, and Alexsandro Santos Soares. Multilayer perceptron and stacked autoencoder for internet traffic prediction. In *IFIP International Conference on Network and Parallel Computing*, pages 61–71. Springer, 2014.

[109] Wenhao Huang, Haikun Hong, Man Li, Weisong Hu, Guojie Song, and Kunqing Xie. Deep architecture for traffic flow prediction. In *International Conference on Advanced Data Mining and Applications*, pages 165–176. Springer, 2013.

[110] Dumitru Erhan, Aaron Courville, and Yoshua Bengio. Understanding representations learned in deep architectures. *Department d'Informatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep*, 1355, 2010.

[111] Eros Pasero and W Moniaci. Artificial neural networks for meteorological nowcast. In *Computational Intelligence for Measurement Systems and Applications, 2004. CIMSA. 2004 IEEE International Conference on*, pages 36–39. IEEE, 2004.

[112] Mario Costa, Walter Moniaci, and Eros Pasero. Info: an artificial neural system to forecast ice formation on the road. In *Computational Intelligence for Measurement Systems and Applications, 2003. CIMSA'03. 2003 IEEE International Symposium on*, pages 216–221. IEEE, 2003.

[113] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[114] Sanam Narejo and Eros Pasero. A hybrid approach for time series forecasting using deep learning and nonlinear autoregressive neural networks. In *INTELLI 2016, The Fifth International Conference on Intelligent Systems and Applications*, pages 69–75. IARIA, 2016.

[115] Francisco S de Albuquerque Filho, Francisco Madeiro, Sérgio MM Fernandes, Paulo SG de Mattos Neto, and Tiago AE Ferreira. Time-series forecasting of pollutant concentration levels using particle swarm optimization and artificial neural networks. *Química Nova*, 36(6):783–789, 2013.

[116] Olga Valenzuela, Ignacio Rojas, Fernando Rojas, Héctor Pomares, Luis Javier Herrera, Alberto Guillén, Luisa Marquez, and Miguel Pasadas. Hybridization of intelligent techniques and arima models for time series prediction. *Fuzzy Sets and Systems*, 159(7):821–845, 2008.

[117] Cagdas Hakan Aladag, Erol Egrioglu, and Cem Kadilar. Forecasting nonlinear time series with a hybrid methodology. *Applied Mathematics Letters*, 22(9):1467–1470, 2009.

[118] Juan J Flores, Roberto Loaeza, Hector Rodriguez, Federico Gonzalez, Beatriz Flores, and Antonio Terceno Gomez. Financial time series forecasting using a hybrid neural-evolutive approach. In *Proceedings of the XV SIGEF International Conference*, pages 547–555, 2009.

[119] Ina Khandelwal, Ratnadip Adhikari, and Ghanshyam Verma. Time series forecasting using hybrid arima and ann models based on dwt decomposition. *Procedia Computer Science*, 48:173–179, 2015.

[120] Gonzalo Acuña, Cristián Ramirez, and Millaray Curilem. Comparing narx and narmax models using ann and svm for cash demand forecasting for atm. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6. IEEE, 2012.

[121] Alexander Hornstein and Ulrich Parlitz. Bias reduction for time series models based on support vector regression. *International Journal of Bifurcation and Chaos*, 14(06):1947–1956, 2004.

[122] A Ya Kaplan and Sergei L Shishkin. Application of the change-point analysis to the investigation of the brain's electrical activity. In *Non-parametric statistical diagnosis*, pages 333–388. Springer, 2000.

[123] Michal Teplan. Fundamentals of eeg measurement. *Measurement science review*, 2(2):1–11, 2002.

[124] Jiajie Zhang and Vimla L Patel. Distributed cognition, representation, and affordance. *Cognition Distributed: How Cognitive Technology Extends Our Minds*, 16:137, 2008.

[125] Sven Hoffmann and Michael Falkenstein. The correction of eye blink artefacts in the eeg: a comparison of two prominent methods. *PLoS One*, 3(8):e3004, 2008.

[126] Gabriele Gratton, Michael GH Coles, and Emanuel Donchin. A new method for off-line removal of ocular artifact. *Electroencephalography and clinical neurophysiology*, 55(4):468–484, 1983.

[127] Leor Shoker, Srreici Sanei, and Mohamed A Latif. Removal of eye blinking artifacts from eeg incorporating a new constrained bss algorithm. In *Sensor Array and Multichannel Signal Processing Workshop Proceedings, 2004*, pages 177–181. IEEE, 2004.

[128] V Krishnaveni, S Jayaraman, L Anitha, and K Ramadoss. Removal of ocular artifacts from eeg using adaptive thresholding of wavelet coefficients. *Journal of Neural Engineering*, 3(4):338, 2006.

[129] Rajesh Singla, Brijil Chambayil, Arun Khosla, and Jayashree Santosh. Comparison of svm and ann for classification of eye events in eeg. *Journal of Biomedical Science and Engineering*, 4(01):62, 2011.

[130] Abdelkader Nasreddine Belkacem, Hideaki Hirose, Natsue Yoshimura, Duk Shin, and Yasuharu Koike. Classification of four eye directions from eeg signals for eye-movement-based communication systems. *life*, 1:3, 2014.

[131] Mohammad Reza Haji Samadi, Zohreh Zakeri, and Neil Cooke. Vog-enhanced ica for removing blink and eye-movement artefacts from eeg. In *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 603–606. IEEE, 2016.

[132] Ting Wang, Sheng-Uei Guan, Ka Lok Man, and TO Ting. Eeg eye state identification using incremental attribute learning with time-series classification. *Mathematical Problems in Engineering*, 2014, 2014.

[133] Ling Li. The differences among eyes-closed, eyes-open and attention states: an eeg study. In *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pages 1–4. IEEE, 2010.

[134] Oliver Rösler and David Suendermann. A first step towards eye state prediction using eeg. *Proc. of the AIHLS*, 2013.

[135] Andrew Frank and Arthur Asuncion. Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of Information and Computer Science*, 213, 2010.

[136] Ting Wang, Sheng Uei Guan, Ka Lok Man, and TO Ting. Time series classification for eeg eye state identification based on incremental attribute learning. In *Computer, Consumer and Control (IS3C), 2014 International Symposium on*, pages 158–161. IEEE, 2014.

[137] Cameron R Hamilton, Shervin Shahryari, and Khaled M Rasheed. Eye state prediction from eeg data using boosted rotational forests. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 429–432. IEEE, 2015.

[138] YM Kim, CY Lee, and CG Lim. Computing intelligence approach for an eye state classification with eeg signal in bci. In *Software Engineering and Information Technology: Proceedings of the 2015 International Conference on Software Engineering and Information Technology (SEIT2015)*, page 265. World Scientific, 2015.

[139] Mridu Sahu, NK Nagwani, Shrish Verma, and Saransh Shirke. An incremental feature reordering (ifr) algorithm to classify eye state identification using eeg. In *Information Systems Design and Intelligent Applications*, pages 803–811. Springer, 2015.

[140] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[141] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Two distributed-state models for generating high-dimensional time series. *Journal of Machine Learning Research*, 12(Mar):1025–1068, 2011.

[142] Souhaib Ben Taieb, Rob J Hyndman, et al. Boosting multi-step autoregressive forecasts. In *ICML*, pages 109–117, 2014.

[143] Anders Bredahl Kock, Timo Teräsvirta, et al. Forecasting with nonlinear time series models. *Oxford Handbook of Economic Forecasting*, pages 61–88, 2011.

[144] Nazim Osman Bushara and Ajith Abraham. Using adaptive neuro-fuzzy inference system (anfis) to improve the long-term rainfall forecasting. *Journal of Network and Innovative Computing*, 3(2015):146–158.

[145] Kin C Luk, James E Ball, and Ashish Sharma. An application of artificial neural networks for rainfall forecasting. *Mathematical and Computer modelling*, 33(6):683–693, 2001.

[146] M Nasseri, Keyvan Asghari, and MJ Abedini. Optimized scenario for rainfall forecasting using genetic algorithm coupled with artificial neural network. *Expert Systems with Applications*, 35(3):1415–1421, 2008.

[147] Wei-Chiang Hong and Ping-Feng Pai. Potential assessment of the support vector regression technique in rainfall forecasting. *Water Resources Management*, 21(2):495–513, 2007.

[148] John Abbot and Jennifer Marohasy. Application of artificial neural networks to rainfall forecasting in queensland, australia. *Advances in Atmospheric Sciences*, 29(4):717–730, 2012.

[149] Nazim Osman Bushara and Ajith Abraham. Computational intelligence in weather forecasting: A review. *Journal of Network and Innovative Computing*, 1(2013):320–331, 2013.

[150] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[151] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[152] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.